GUIDA PER L'APPLE

VOLUME 1

L'APPLE STANDARD

Vincent Chaix

Benoît de Merly

Edimicro

Guida per l'APPLE

1. L'APPLE standard

DELLO STESSO EDITORE

Volumi pubblicati

D.-J. David - La scoperta del Commodore 64
D.-J. David - La pratica del Commodore 64
X. Linant de Bellefonds - La pratica dello ZX Spectrum - Vol. 1
M. Henrot - La pratica dello ZX Spectrum - Vol. 2
J.-F. Séhan - Chiavi per lo ZX Spectrum
C. Galais - Vademecum per Applesoft
J. Boisgontier - L'Apple e i suoi files

Volumi di prossima pubblicazione

J. Boisgontier - 102 programmi per il Commodore 64
J. Boisgontier - Commodore 64: metodi pratici
J. Boisgontier, S. Brebion, G. Foucault - Il Commodore 64 per tutti
B. De Merly - Guida per l'Apple - Vol. 2
B. De Merly - Guida per l'Apple - Vol. 3
F. Lévy - Esercizi per l'Apple II
J. Boisgontier - 36 programmi su Apple II per tutti
F. Lévy - Esercizi per lo ZX Spectrum
J.-F. Séhan - Alla ribalta: lo ZX Spectrum
J. Deconchat - 102 programmi per ZX Spectrum e ZX 81
D.A. Lien - Dizionario del Basic
J. Boisgontier - Il Basic per tutti
A. Pinaud - CP/M passo dopo passo

BENOÎT DE MERLY

Guida per l'APPLE

1. L'APPLE standard

Edizione italiana a cura di
FRANCO POTENZA

Traduzione di
ALBERT PALACI

Editsi-Editoriale per le scienze informatiche S.r.l.
MILANO 1984

Prefazione

Quando lanciarono sul mercato questa piccola scatola color crema che essi avevano battezzato Apple II, Steve Jobs e i suoi Colleghi erano lontani dal rendersi conto dell'avvenire che attendeva il primo personal computer.
Certamente alcuni dei loro amici si erano dimostrati entusiasti di fronte alle possibilità dell'apparecchio. Ma alla fine del 1977 chi poteva prevedere un tale successo? Alcuni anni più tardi un milione di esemplari erano entrati negli uffici, nelle fabbriche, nelle scuole e nelle case di tutto il mondo.
Quando è apparsa la parola microcalcolatore i più lungimiranti immaginavano già dei grandi computer ridotti a una dimensione molto piccola, ma nessuno avrebbe pensato che voi, io e i nostri ragazzi avremmo rapidamente potuto adoperarli per giocare, studiare e lavorare.
Steve Jobs aveva una certa idea di che cosa doveva essere il suo Apple II. Gli utilizzatori hanno moltiplicato questa idea: mettendo il naso nelle istruzioni, provando comandi diversi e diversi modi di impiego, gli utenti hanno di fatto inventato la microinformatica.
Per me che devo spesso presentare questa nuova disciplina al vasto pubblico televisivo poco abituato a queste macchine, la differenza tra microinformatica e Informatica (notate la maiuscola) è almeno pari a quella tra una Formula uno e una utilitaria. È possibile certamente con un po' di allenamento e molti soldi pilotare una macchina da corsa su un circuito rapido, il che equivale, nel nostro campo, a padroneggiare le grosse macchine dei centri di calcolo.
Se, più semplicemente, voi amate il vostro mestiere, se volete migliorare la vostra vita quotidiana in ufficio e a casa, sappiate che la microinformatica è alla vostra portata soprattutto grazie a strumenti come questo che avete in mano.
Questa guida si mostrerà presto indispensabile nella ricerca delle migliori possibilità del vostro Apple. Grazie ai diversi livelli di comprensione potrete affrontare in una prima lettura solo i capitoli semplici, i capitoli sul Basic per esempio. Più ne saprete e più avrete voglia di andare avanti.
Voi diventerete colui o colei che ne sa un po' di più e che comprende e padroneggia quindi meglio la sua macchina. Le vostre possibilità saranno decuplicate e vi apparirà in pieno il beneficio del vostro investimento: è appunto questo lo scopo di questo libro.

GEORGES LECLERE

Marzo 1983

Sommario

APPENDICI

CAPITOLO 1

Presentazione dell'Apple II

1.1 INTRODUZIONE

L'Apple II è un microelaboratore utilizzabile dagli hobbisti e dai professionisti. Esso si presenta in un contenitore compatto, facilmente trasportabile. La tastiera, incorporata sul lato anteriore del contenitore, possiede le seguenti caratteristiche:

- 63 tasti che consentono di generare le 128 configurazioni del codice ASCII;
- caratteri minuscoli e maiuscoli;
- disposizione dei tasti QZERTY (macchina da scrivere italiana), oppure QWERTY selezionabile mediante un interruttore.

L'uscita video si trova nella parte posteriore dell'apparecchio ed emette dei segnali secondo il sistema PAL.
La figura A mostra un Apple II munito di una unità per dischetti e di un video.
Come indica la figura B, premete verso l'alto, al centro del lato posteriore, onde poter sollevare il coperchio.
A questo punto potete distinguere in figura C i seguenti elementi:

- il blocco autonomo di alimentazione che genera tensioni continue comprese tra —5V e +5V, oppure tra —12V e +12V;
- la scheda principale contenente il microprocessore 6502, sulla quale ci soffermeremo nel paragrafo successivo;
- l'altoparlante che consente di generare dei suoni di durata e di frequenza variabili, posto al di sotto della tastiera (cfr. cap. 3);
- l'interfaccia video e il modulo PAL;
- l'interfaccia per cassette;
- una presa per manopole di giochi o per dispositivi d'ingresso-uscita analogici;

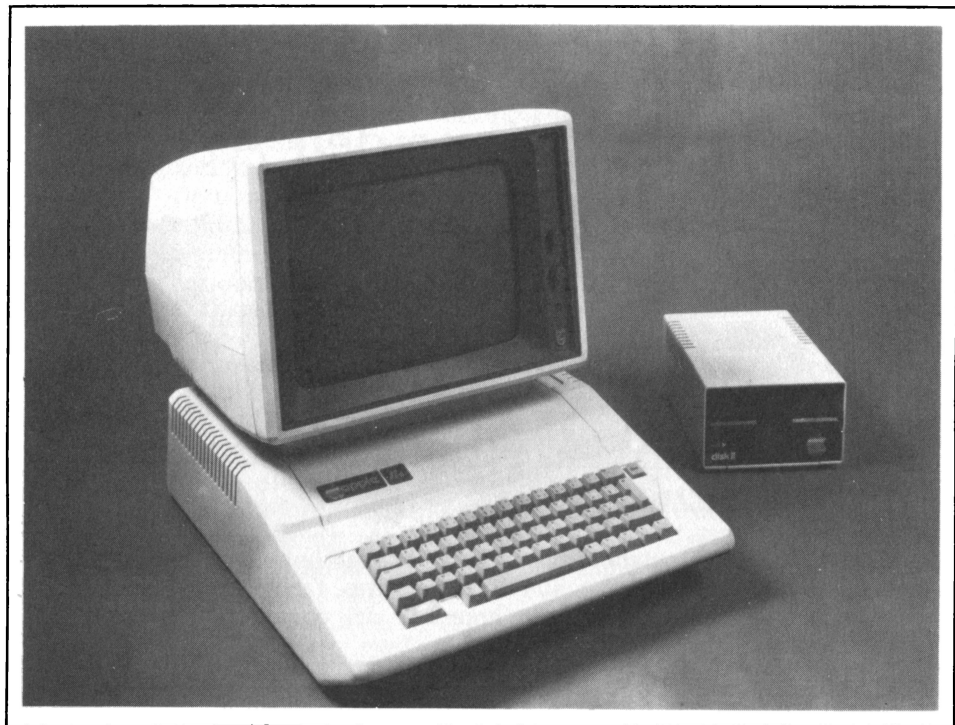



Foto A: configurazione Apple II e.

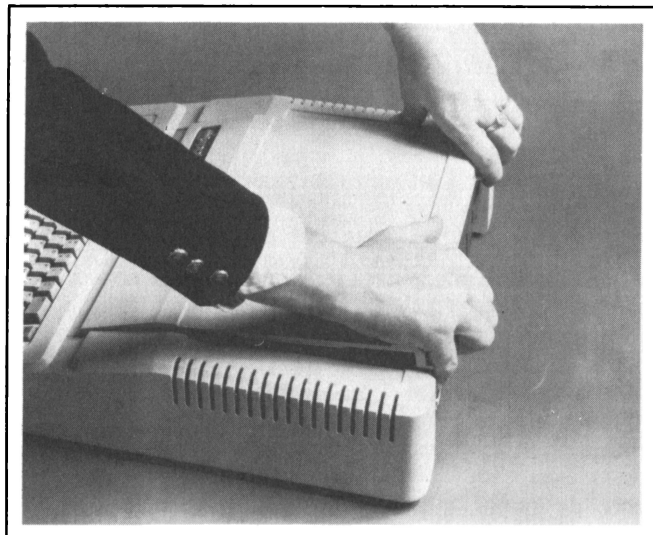



Foto B: rimozione del coperchio.

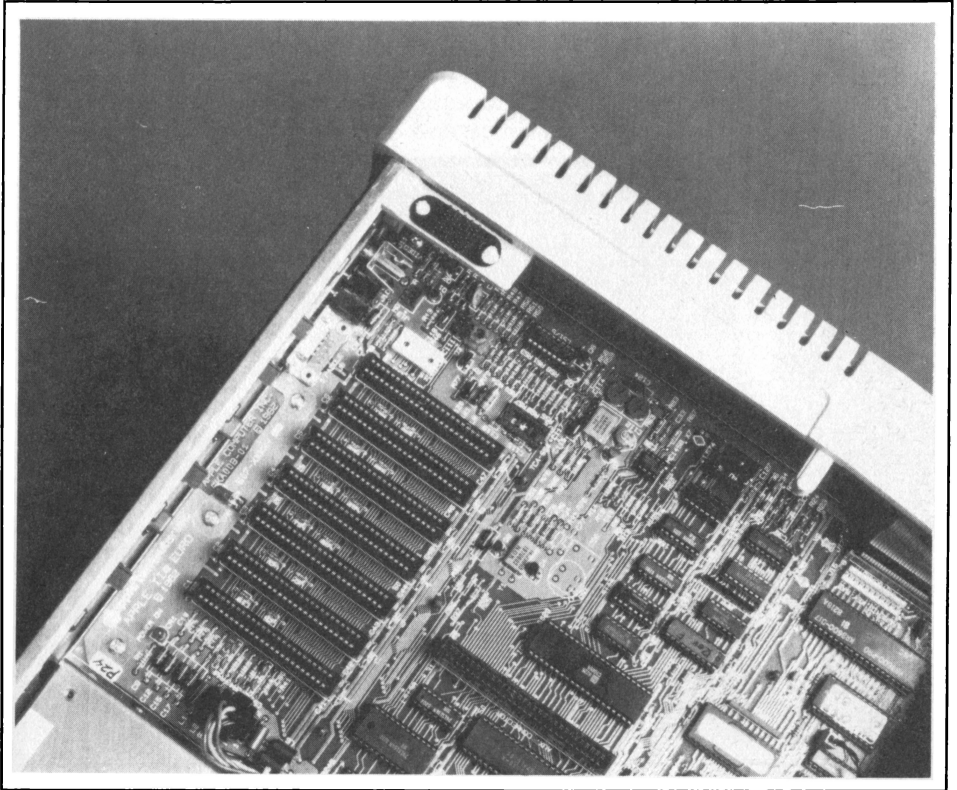



Foto C: vista interna dell'Apple II e.

- 7 connettori che consentono l'inserimento di schede aggiuntive per permettere all'Apple lo sviluppo di nuove funzioni;
- il pannello posteriore concepito per montaggi e smontaggi rapidi;
- un connettore ausiliario destinato alle schede video.

*La scheda di gestione della tastiera dell'*Apple II Plus *è stata integrata nella scheda primitiva dell'*Apple II e, *che comprende inoltre un connettore per una tastiera numerica separata.*

1.2 UNITÀ CENTRALE

Esaminiamo ora più da vicino la scheda principale dell'Apple IIe.
In figura D, compaiono i seguenti elementi:

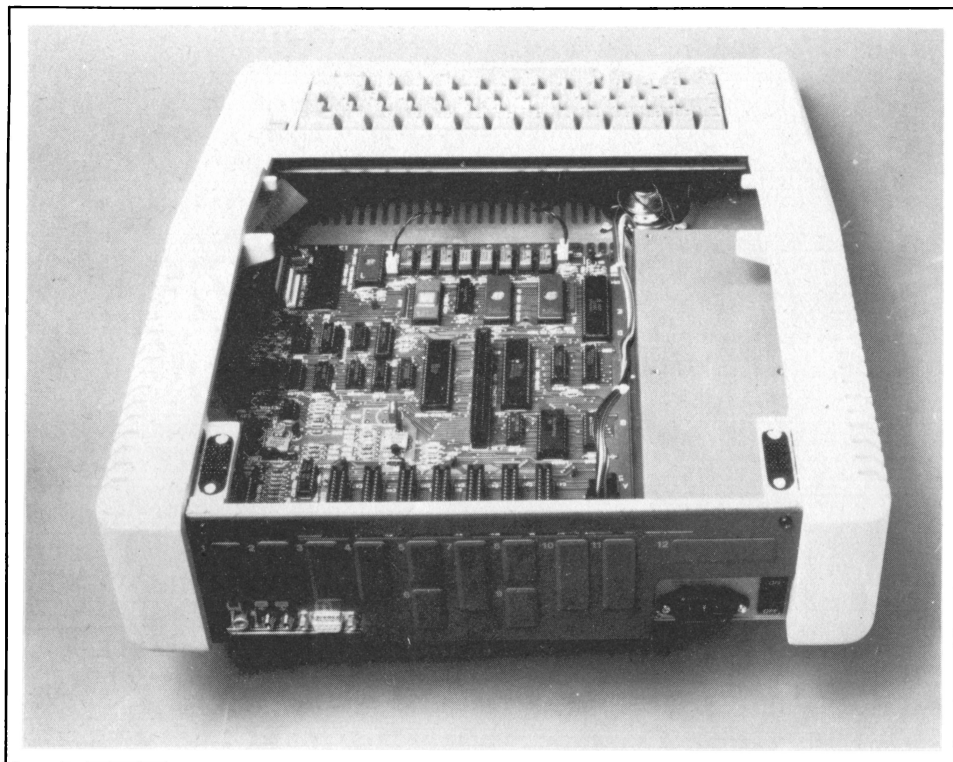



Foto D: unità centrale.

- Il microprocessore 6502A, elemento centrale del sistema, il quale compie operazioni di calcolo, esegue i programmi, conversa con la memoria centrale e le periferiche ecc. Esso funziona a 1MHz e consente di effettuare fino a 500.000 operazioni al secondo su 8 bits.
- 2 integrati di memoria ROM (8K × 8bits) contenenti nei 16K bytes il Basic Applesoft, il programma Monitor di cui parleremo nel capitolo 4 e che gestisce le operazioni di ingresso-uscita e i programmi di autocontrollo del sistema.
- 8 integrati di memoria RAM (64K × 1bit) cancellabili con l'interruzione della corrente e che offrono 64K bytes ai programmi utenti.
- Un'unità di gestione della memoria che consente di accedere di volta in volta alle memorie RAM e ROM.

Supporremo nel seguito del manuale che la ROM Basic *sia stata selezionata, il che consente di avere contemporaneamente* 48K *bytes di* RAM *a disposizione.*

1.3 SCHEDE PER ESPANSIONI CLASSICHE

Sul fondo della scheda centrale si trovano dei connettori che consentono di aggiungere delle schede di espansione. Descriviamo qui di seguito le schede che permettono il cambiamento del sistema di utilizzo o del linguaggio.
In primo luogo citiamo la Language Card che consente di ottenere il sistema

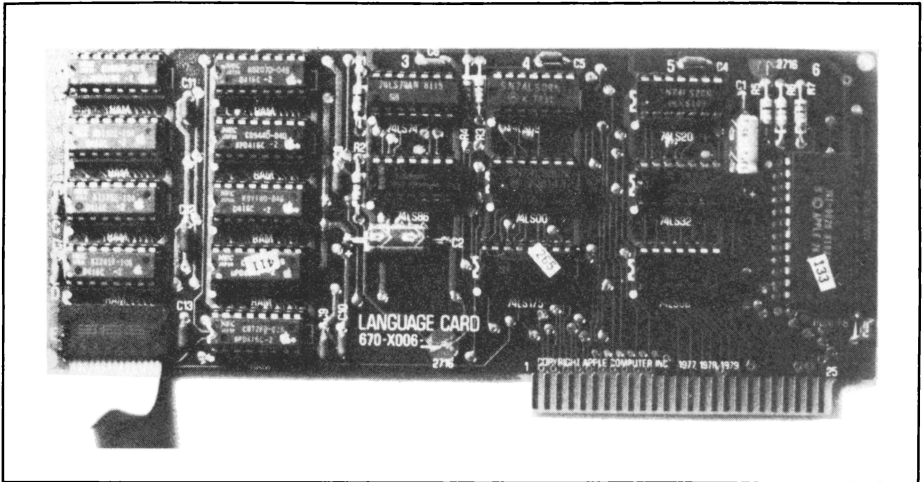



Foto E: Language Card Apple

d'uso P-SYSTEM U.C.S.D. e i linguaggi Pascal, Fortran, Logo ecc. (cfr. cap. 2 vol 2).
Questa scheda è stata integrata nel modello Apple II e *ed è sufficiente avere i*

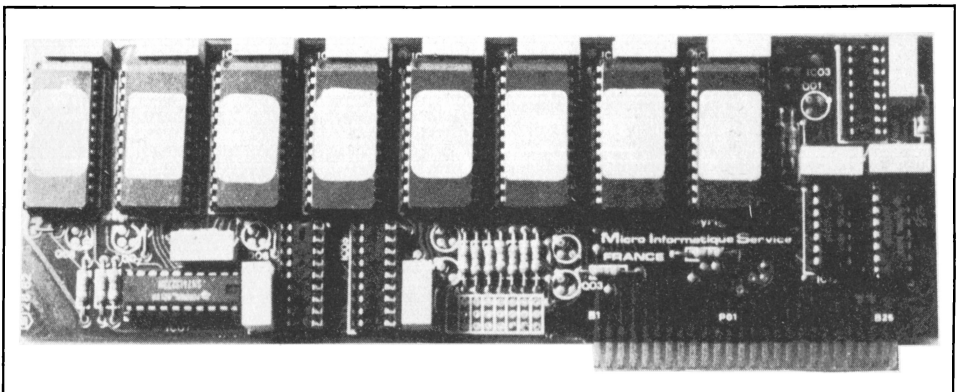



Foto F: scheda M/DOS.

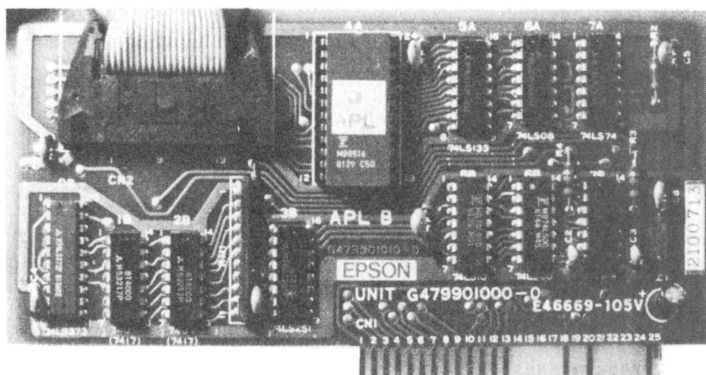



Foto G: scheda di interfaccia parallela Epson.

corrispondenti dischetti per poter disporre dei linguaggi Pascal, Fortran, Logo ecc.
Secondo la stessa logica, la scheda M/DOS, creata dalla società MIS, offre un sistema d'uso molto più efficiente del DOS Apple, con gestione di registri di mascheratura per ingresso-uscita, flussi sequenziali, a indice a più chiavi ecc.
Tra le altre schede classiche, due tipi di scheda devono essere citati:

- scheda di interfaccia parallela, che consente la comunicazione con una stampante;
- scheda di interfaccia seriale, che consente la comunicazione con un terminale dotato di schermo e tastiera, con una stampante, con un'altra macchina oppure con un canale teleinformatico;
- scheda modem, che consente di comunicare a distanza con un altro calcolatore, tramite linea telefonica.

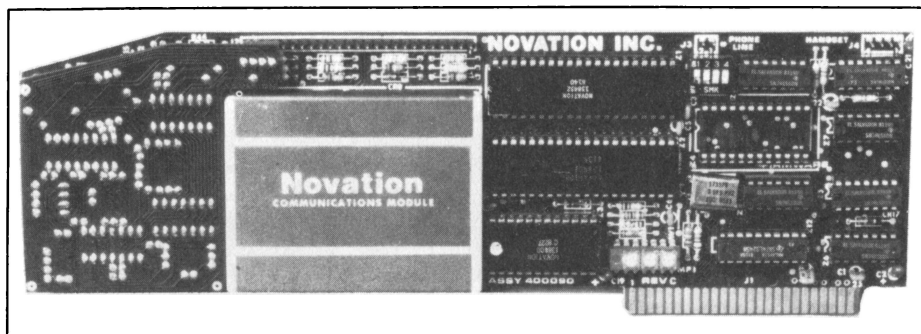



Foto H: scheda modem.

1.4 OUTPUT TELEVISORE E VARIANTI

L'Apple II permette di selezionare la funzione testo, la funzione grafica a bassa risoluzione e, quando è collegato ad un video a colori o ad un televisore a colori (con opzione RVB PERITEL), la funzione grafica a colori. I comandi di grafica permettono di selezionare una delle due "pagine" disponibili sullo schermo con o senza le quattro righe di testo nella parte inferiore dello schermo.
Modi di rappresentazione (matrice 5×7):

- testo su 40 colonne (televisore o video);
- testo su 80 colonne con scheda opzionale e video;
- grafici a colori a bassa risoluzione (video o televisore con scheda RVB PERITEL);
- grafici a risoluzione molto elevata (scheda a 80 colonne con in aggiunta 64K bytes di memoria).

Capacità del testo:

- 24 righe di 40 colonne;
- 24 righe di 80 colonne con scheda opzionale.

Campo dei caratteri:

- 96 caratteri ASCII stampabili (maiuscole e minuscole);
- caratteri italiani ed inglesi.

Formati di rappresentazione:

- normale, inversa, lampeggiante;
- grafici a bassa risoluzione:
  - 16 colori;
  - risoluzione: # 40 orizzontale × 48 verticale,
    # 40 orizzontale × 40 verticale con quattro righe di testo;
- grafici ad alta risoluzione:
  - 6 colori: nero, bianco, verde, blu, viola, arancione;
  - risoluzione: # 280 orizzontale × 192 verticale;
    # 280 orizzontale × 160 verticale con quattro righe di testo;
- grafici a risoluzione molto elevata: 560 orizzontale × 192 verticale.

Le schede di espansione video devono essere inserite nel connettore ausiliario, situato al centro della scheda madre.

1.5 ALTRE ESPANSIONI POSSIBILI

La forza commerciale dell'Apple II risiede nel fatto che, mediante una scheda di espansione, il comportamento della macchina può essere completamente modificato.
L'esempio migliore che vi possiamo offrire riguarda le schede che concerno-

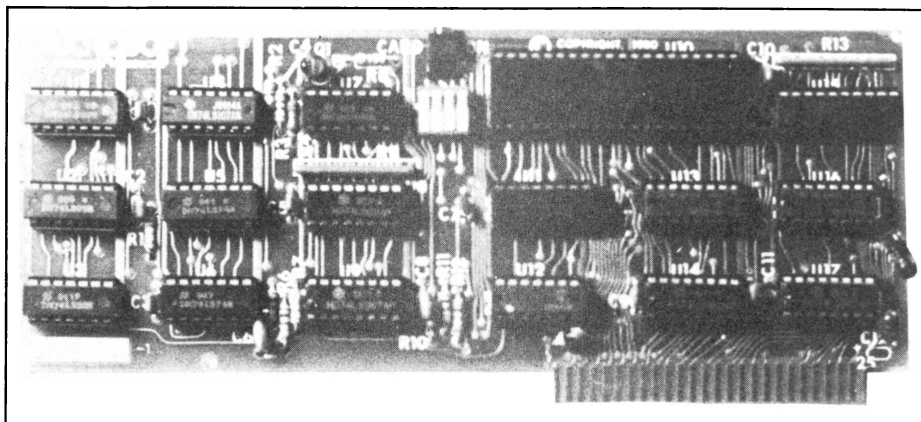



Foto I: Softcard Z80

no altri processori (Z80, 6809, 8088, ...), o altre schede che consentono di ottenere la sintesi della voce, il trattamento di immagini video o che trasformano il sistema in oscilloscopio.
Esistono inoltre cinque tipi di schede di espansione non classici:

- Le schede di espansione di memoria di 16K, 32K, 64K e anche di 128K bytes. Esempio: la scheda RAM del Microsoft o la scheda Legend 128K che può essere utilizzata come simulatrice di una unità per dischetti.
- Le schede "processore" (Z80, 6809, 8088) alle quali si riferisce la Softcard Z80 descritta nel volume 2.
- Le schede di interfaccia con apparecchiature esterne (Bus IEEE/488, strumenti di misura, convertitore analogico-digitale e digitale-analogico, oscilloscopio, ...).
- Schede per il trattamento della voce.
- Schede per la digitalizzazione del mondo esterno.

Daremo nel volume 2 un'ampia panoramica delle schede di espansione disponibili dell'Apple II.

CAPITOLO 2

Il Basic Applesoft

2.1 INTRODUZIONE

Il Basic Applesoft è presente nella memoria ROM nell'Apple II e e nell'Apple II Plus. Questo capitolo ha dunque per oggetto la sintassi del linguaggio e l'impiego di una stampante. Le possibilità grafiche saranno affrontate nel prossimo capitolo.

2.2 DESCRIZIONE SINTATTICA DELL'APPLESOFT

2.2.1 Variabili semplici

Esistono i seguenti tipi di variabili semplici:

Tipo	Nome di variabile	Valori possibili
Intero	AB%	+/— 32767
Reale	AB	+/— 9.99999999E +37
Stringa di caratteri	AB$	0–255 caratteri alfanumerici

Ove A *rappresenta una lettera e* B *una lettera o un numero.*

Sono presi in considerazione solamente, oltre al tipo, i due primi caratteri del nome di una variabile.
Ad esempio: PR1% e PR2% corrispondono alla stessa variabile ma PR1% e PR2$ sono variabili diverse in quanto i tipi sono differenti.
L'istruzione CLEAR azzera tutte le variabili siano esse semplici o no.

2.2.2 Tabelle

Ritroviamo i tre tipi di variabili:

Tipo	Nome di variabile	Dichiarazione
Intero	AB% (4,6,3)	DIM AB% (5,6,7)
Reale	AB (2,3)	DIM AB (2,4)
Stringa di caratteri	AB$ (2,8,6)	DIM AB$ (2,8,6)

La dimensione delle tabelle è limitata esclusivamente dalla dimensione della memoria disponibile. Prima di utilizzare una tabella, è necessario dimensionarla mediante l'istruzione DIM (cfr. la dichiarazione in tabella precedente).
L'indice che viene adoperato in una tabella deve essere un numero compreso tra 0 e il valore indicato nella dichiarazione. Ad esempio la tabella T1 (2,3,4) è costituita da (2 + 1)*(3 + 1)*(4 + 1) = 60 elementi diversi.
Il numero di dimensioni di una tabella è variabile da 1 a 88.
La lunghezza degli elementi di una variabile "stringa di caratteri" in una tabella va da 0 *a* 255 *caratteri.*
Se non viene specificato il numero di elementi (default), il Basic considererà una tabella non dimensionata come avente 11 elementi per dimensione.

2.2.3 Operatori numerici

Questi operatori vengono trattati secondo un ordine di priorità decrescente:

( )	parentesi di delimitazione di espressione
+ —	operatori unari (* + 1, *—1)
^	elevamento a potenza
*	prodotto
/	divisione
+	somma
—	sottrazione

2.2.4 Operatori di relazione

La loro priorità, inferiore a quella degli operatori numerici, viene stabilita secondo il seguente ordine decrescente:

>	strettamente maggiore di
<	strettamente minore di
> =	maggiore o uguale a
< =	minore o uguale a
= >	maggiore o uguale a
= <	minore o uguale a
< >	diverso da
> <	diverso da
=	uguale a

Le stringhe di caratteri possono essere confrontate tra loro secondo la rappresentazione in codice ASCII *dei caratteri in esse contenuti.*

2.2.5 Operatori logici

Esistono tre tipi di operatori logici:

NOT negazione logica
(priorità immediatamente superiore all'elevamento a potenza)
AND prodotto logico
OR somma logica
(i due precedenti operatori sono di priorità inferiore)

Il risultato di un confronto o di un'operazione logica può assumere due valori: vero o falso.
In seguito, indicheremo con il termine "espressione" una serie di operazioni tra variabili e costanti reali, o intere, oppure tra stringhe di caratteri.

Esempio: (AB^E) < (D–E) AND (A + 1 > B)

Denoteremo con X, Y e Z le espressioni numeriche mentre con VR e VI indicheremo le variabili reali o intere.

2.2.6 Istruzioni di controllo

Le istruzioni seguenti permettono di eseguire dei cicli, oppure dei salti calcolati, condizionati o incondizionati, oppure consentono l'utilizzo di sottoprogrammi.

```
FOR VR=2 TO 10 STEP 4  )
...                    } (ciclo di programma)
...                    }
NEXT VR                )
```

1. VR *rappresenta, per le istruzioni* FOR *e* NEXT, *una variabile reale. Gli altri termini dell'istruzione* FOR *sono delle espressioni numeriche reali o no.*
2. *Le istruzioni comprese nel ciclo vengono eseguite almeno una volta, perché il test ha luogo in corrispondenza dell'istruzione* NEXT.

IF <espressione> THEN istruzioni	Esecuzione delle istruzioni indicate o salto alla riga indicata se viene verificata la condizione
ON X GOTO L1,L2	Salto calcolato alla riga L1 se l'espressione X vale 1
GOTO n.riga	Salto incondizionato alla riga indicata
ON X GOSUB S1,S2	Chiamata a sottoprogramma calcolata

GOSUB S	Chiamata a sottoprogramma posto alla riga S
RETURN	Ritorno da sottoprogramma
POP	Soppressione di un livello di sottoprogramma (equivalente a RETURN ma con innesco dell'istruzione successiva alla POP)
STOP	Arresto del programma e stampa del numero di riga in corrispondenza di STOP
END	Arresto normale del programma
RUN	Lancio dell'esecuzione di un programma
CTRL-C oppure RESET	Arresto del programma causato dall'utente da tastiera
CONT	Ripresa di un programma dopo STOP, END, CTRL-C

2.2.7 Accesso alla memoria e chiamata di sottoprogramma assemblatore

POKE X,Y	Posizionamento all'indirizzo X del byte Y
PEEK (X)	Lettura del valore del byte all'indirizzo X

Le istruzioni POKE e PEEK vi saranno di grande aiuto per elaborare programmi complessi. Potrete farne a meno per programmi semplici.

FRE (0)	Lettura di un numero di bytes non utilizzati in memoria e riorganizzazione della zona di memoria occupata dalle variabili
HIMEM	Posizionamento dell'indirizzo massimo utilizzabile in memoria dal Basic
LOMEM	Posizionamento dell'indirizzo minimo utilizzabile in memoria dal Basic

Se desiderate servirvi delle possibilità grafiche del vostro sistema, oppure realizzare dei sottoprogrammi in assembler, vi consigliamo di limitare lo spazio occupato dal Basic e dai suoi programmi onde non rischiare di sporcare la memoria.

CALL X	Chiamata a sottoprogramma in assembler situato all'indirizzo indicato dall'espressione X
USR (X)	Chiamata della funzione assembler con trasmissione del parametro uguale al valore dell'espressione X

Al momento della chiamata della funzione USR, il Basic converte il parametro in un numero reale, lo pone nell'accumulatore flottante posto agli indirizzi 157–163 e effettua un salto all'indirizzo 10 che deve contenere un salto al sottoprogramma assembler.

WAIT X,Y,Z	Ciclo d'attesa finché l'espressione (X) XOR Z AND Y viene verificata.

(X) *indica il byte all'indirizzo* X

&	Salto all'indirizzo $3F5 (equivalente Basic della funzione CTRL–Y del programma di Monitor, cfr. pag. 4)

2.2.8 TEXT–EDITING sullo schermo

CTRL–X	Cancellazione della riga in corso di battitura
Tasto freccia a destra	Ricopiatura in memoria del carattere posizionato sotto il cursore e spostamento del cursore di una posizione verso destra
Tasto freccia a sinistra	Cancellazione di un carattere e spostamento del cursore di una posizione verso sinistra
ESC A	Spostamento del cursore di una posizione verso destra
ESC B	Spostamento del cursore di una posizione verso sinistra
ESC C	Spostamento del cursore di una posizione verso il basso
ESC D	Spostamento del cursore di una posizione verso l'alto

Per quanto riguarda le ultime quattro funzioni, il carattere posizionato sotto il cursore non viene ricopiato in memoria.

ESC E	Cancellazione dei caratteri situati tra il cursore e la fine della riga
ESC F	Cancellazione dei caratteri situati tra il cursore e il resto dello schermo
ESC §	Cancellazione dell'intero schermo

Esempio: trasformazione della riga

10 PRINT "APPLE" in
10 PRINT "MELA"

Procedimento: 1. Ricopiatura di 10 PRINT"
a. posizionare il cursore sull'1 del numero 10
b. premere 11 volte il tasto freccia a destra

2. Cancellazione di APPLE
   premere 5 volte ESC A
3. Scrittura di MELA
   a. posizionarsi sulla riga superiore
      premere ESC D
   b. scrivere MELA
   c. posizionarsi sulle secondo "
      premere 4 volte ESC B
   d. convalidare le "
      premere il tasto freccia a destra
   e. fine della modifica
      premere RETURN

2.2.9 Istruzioni di ingresso-uscita sullo schermo

VTAB X	Posizionamento del cursore alla riga X (con X da 1 a 24)
HTAB X	Posizionamento del cursore alla colonna X (con X da 1 a 255)

La riga logica gestita dall'istruzione HTAB *corrisponde a più righe fisiche sullo schermo. Ad esempio con* X *tra* 1 *e* 40 *ci si posiziona sulla riga corrente, con* X *tra* 41 *e* 80 *ci si posiziona sulla riga seguente, e così di seguito.*

POS (0)	Segnalazione della posizione del cursore sullo schermo	
HOME	Cancellazione dello schermo e posizionamento del cursore nell'angolo superiore sinistro	
NORMAL INVERSE FLASH	Display dei caratteri sullo schermo secondo la funzione scelta	(bianco su nero) (nero su bianco) (lampeggiante)
SPEED = X	Modifica della velocità di display dei caratteri sullo schermo	
GET A$	Acquisizione di un carattere alfanumerico da tastiera e sua assegnazione alla relativa stringa senza che debba essere premuto il tasto RETURN	
INPUT A%	Display di un punto interrogativo e acquisizione di una variabile (numerica o no) dopo la pressione del tasto RETURN	

INPUT "X,Y,Z";A,B%,C$	Display della stringa di caratteri indicata senza punto interrogativo e acquisizione delle variabili da tastiera
PRINT "A = ";A,"B > ";C "D< = ";B	Display sullo schermo delle stringhe "A = ", "B> = ", "D< = " e dei valori delle variabili A,C,B con il seguente formato: A = valore, B> = valore, D< = valore

I punti e virgola concatenano i campi, mentre le virgole li posizionano al principio di una zona di tabulazione di 16 *caratteri (cioè alle colonne* 1, 17, 32). *Un punto e virgola posto alla fine di un'istruzione* PRINT *evita il salto alla riga seguente.*

Esempi:

	colonna 1	**colonna 17**	**colonna 32**
PRINT"XYZ","2";3	XYZ	23	
PRINT"X";"Y";"Z2",3	XYZ2	3	
PRINT"X","Y";"Z",32	X	YZ	32

SPC(X)	Questa funzione viene utilizzata in una istruzione PRINT per inserire X spazi nella riga di stampa
TAB(X)	Questa funzione viene utilizzata in una istruzione PRINT per posizionare il cursore alla colonna X (con X tra 0 e 255)

Le istruzioni

10 HTAB 23:PRINT"ABC" e
10 PRINT TAB(22);"ABC"

sono equivalenti poiché HTAB *conta le colonne a partire dal numero* 1, *mentre* TAB *parte dallo* 0>

READ A1,A2%,A1$,... DATA 1.3,2,"erty",...	Lettura dei dati specificati nelle istruzioni DATA e loro assegnamento alle variabili relative
RESTORE	Ripresa della lettura delle DATA a partire dalla prima istruzione DATA del programma

2.2.10 Creazione di una finestra di testo sullo schermo

È possibile ridurre la dimensione dello schermo su cui si visualizza il testo per conservare, ad esempio, un titolo in cima ad una pagina o per simulare delle maschere per acquisizione dati ecc.
A questo scopo, utilizzerete l'istruzione POKE di modifica della memoria con i seguenti indirizzi:

Indirizzo	Effetto	
32	Margine sinistro	(da 1 a 40)
33	Larghezza	(da 1 a 40)
34	Prima riga	(da 1 a 24)
35	Ultima riga	(da 1 a 24)

Le dimensioni della finestra non devono superare quelle dello schermo poiché, se questo avvenisse, la finestra distruggerebbe una zona di memoria che non le è attribuita. È indispensabile che vengano sempre verificate le seguenti relazioni:

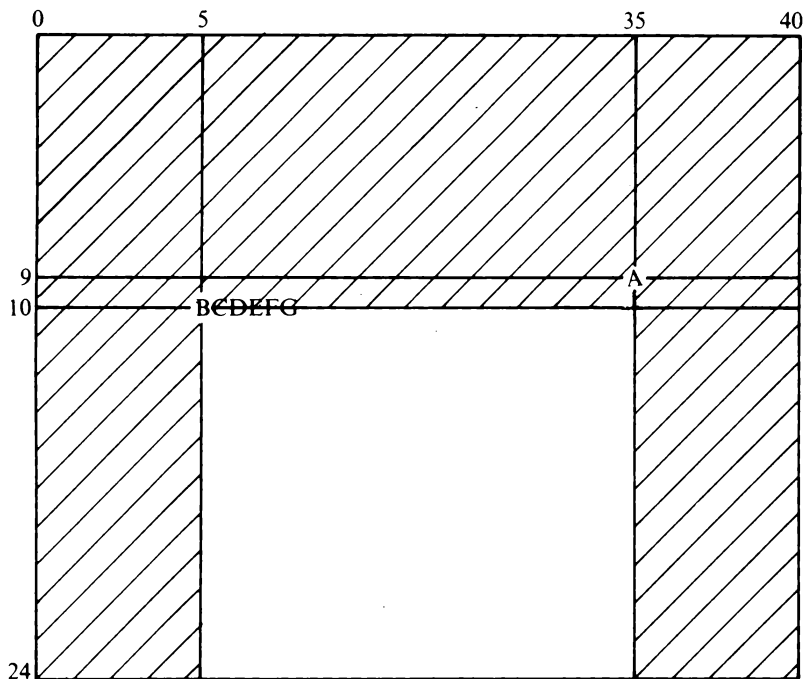



Fig. 2.1

```
PEEK(34) +PEEK(35) < = 24
PEEK(34) < = PEEK(35)
PEEK(32) + PEEK(33) < = 40
```

Ecco il comportamento delle istruzioni d'ingresso-uscita sullo schermo:

- HTAB tiene conto della nuova larghezza dello schermo;
- VTAB non ha alcun effetto;
- PRINT e le funzioni simili (TAB, SPC, ...):
  1. la posizione attuale del cursore non è verificata,
  2. tutti i caratteri sono visualizzati all'interno della finestra.

Ad esempio, se avete posizionato la prima riga a 10, il margine sinistro a 5, la larghezza a 20, otterrete la situazione della figura 2.1 con l'utilizzo della seguente stringa di istruzioni:

```
VTAB 9: HTAB 35: PRINT"ABCDEFG"
```

- INPUT: ha lo stesso comportamento dell'istruzione PRINT.

2.2.11 Istruzioni di ingresso-uscita con le periferiche

Le istruzioni necessarie per lavorare con le periferiche (stampante, DOS 3.3, ...) sono:

IN# numero di connettore	Lettura dei dati provenienti dalla periferica collegata al numero di connettore precisato
PR# numero di connettore	Uscita dei dati verso la periferica collegata al numero di connettore precisato
LOAD	Caricamento di un programma da cassetta
SAVE	Salvataggio di un programma su cassetta
LOAD nome del file	Caricamento di un programma residente su disco
SAVE nome del file	Salvataggio di un programma residente su disco
RUN	Esecuzione di un programma residente su disco

(per quanto riguarda le ultime tre istruzioni cfr. vol. 2: il DOS 3.3)

STORE A	Salvataggio di una tabella numerica su cassetta
RECALL A	Ricaricamento dalla cassetta di una tabella numerica dimensionata in precedenza

PDL(I)	Lettura del valore indicato dalla manopola per giochi numero I (cfr. par. 3.7)

2.2.12 Trattamento degli errori

Nel caso in cui non abbiate scritto nel vostro programma nessuna istruzione di trattamento degli errori, il Basic Applesoft arresterà il programma all'apparire di un errore e visualizzerà il messaggio corrispondente.
Per evitare ciò, voi potete prendere il controllo delle operazioni ponendo all'inizio del vostro programma l'istruzione seguente:

ON ERROR GOTO numero di riga

Incontrando un errore qualsiasi, il Basic effettuerà un salto alla riga indicata. Potrete così disporre delle due seguenti informazioni:

- **Codice dell'errore**

Esso è situato all'indirizzo 222 e può assumere i valori indicati nella seguente tabella:

Codice	**Messaggio di errore**	
0	NEXT without FOR	È stata incontrata un'istruzione NEXT non preceduta dalla relativa FOR
16	Syntax Error	Errore di sintassi
22	RETURN without GOSUB	È stata incontrata un'istruzione RETURN non preceduta da GOSUB
42	Out of DATA	I dati delle istruzioni DATA sono stati completamente utilizzati
53	Illegal Quantity	
69	Overflow	Superamento di capacità
77	Out of memory	Non c'è più memoria libera
90	Undefined Statement	Istruzione (comando) sconosciuto
107	Bad Subscript	L'utilizzo di una tabella non è compatibile con la sua dichiarazione (dimensioni, indice scorretto...)
120	Redimensionned Array	Non è ammesso dimensionare due volte la stessa tabella
133	Division by Zero	Divisione per zero
163	Type mismatch	Impiego di un tipo di variabile
176	String Too Long	Eccessiva lunghezza di una stringa di caratteri (> =256)
191	Formula Too Complex	Il Basic non è in grado di trattare un'espressione così complessa
224	Undefined Function	La funzione utilizzata non è definita
254	Bad Response to INPUT Statement	Risposta scorretta all'istruzione INPUT
255	CTRL-C Interrupt Attempted	Intervento di un CTRL-C

Poiché il DOS *utilizza gli stessi indirizzi, completeremo questa tabella nel volume 2.*

- **Numero della riga in cui l'errore si è verificato**

Per conoscerlo è sufficiente leggere il contenuto degli indirizzi 218 e 219:

L = PEEK(218) + 256*PEEK(219)

Dopo aver terminato il trattamento dell'errore, vi sono offerte due possibilità:

- l'istruzione RESUME che provoca la ripresa del programma dall'istruzione corretta;
- l'istruzione GOTO.

Il compito dell'istruzione ON ERROR GOTO *non è che quello di posizionare il byte d'indirizzo* 216 *ad un valore superiore a* 128. *Il suo effetto può essere annullato dall'istruzione seguente:*

POKE 216,0

Un esempio di utilizzo dell'istruzione ON ERROR GOTO è presentato al paragrafo 3.3.5.

2.2.13 Comandi diversi

NEW	Cancellazione del programma in memoria
LIST[L1,][L2]	Listing delle righe comprese tra L1 e L2
DEL L1,L2	Cancellazione delle righe di etichetta compresa nell'intervallo L1,L2
TRACE	Visualizzazione del numero di riga di un'istruzione in esecuzione
NOTRACE	Eliminazione dell'effetto di TRACE
REM	Istruzione di commento

2.2.14 Funzioni riguardanti le stringhe di caratteri

LEN(A$)	Funzione indicante la lunghezza (numero di caratteri) di una stringa di caratteri
VAL(A$)	Funzione indicante il valore numerico di una stringa di caratteri VAL("23456") = 23456
STR$(X)	Funzione che converte un numero in una stringa di caratteri STR$(2) = "2"
ASC(A$)	Funzione che restituisce il codice ASCII del primo carattere della stringa A$
CHR$(X)	Conversione da codice ASCII ad una stringa di caratteri ASC(CHR$(10)) = 10 CHR$(65) = "A"

LEFT$(A$,X)	Ottenimento della stringa dei primi X caratteri di A$
RIGHT$(A$,X)	Ottenimento della stringa degli ultimi X caratteri di A$
MID$(A$,X,Y)	Ottenimento della stringa degli Y caratteri estratti a partire dall'Xesimo carattere di A$ MID$("ABCDE",3,2) = "CDE"
+	Concatenazione di stringhe di caratteri

2.2.15 Funzioni matematiche

DEF FN A(X) = 3*X + 4   Definizione della funzione FNA

FUNZIONI TRIGONOMETRICHE

SIN(X)	Seno di X (in radianti)
COS(X)	Coseno di X (in radianti)
TAN(X)	Tangente di X (in radianti)
ATN(X)	Arcotangente di X

GENERAZIONE DI UN NUMERO CASUALE

RND(X)	Restituisce un numero casuale compreso tra 0 e 1

* *Se* X *è positivo, ad ogni chiamata della funzione viene generato un nuovo numero casuale.*
* *Se* X *è nullo, viene generato nuovamente l'ultimo numero casuale.*
* *Se* X *è negativo, viene generato il numero corrispondente a* X *(tabella dei numeri in memoria).*

RND(—3) = 4.48217179E-08

ALTRE FUNZIONI MATEMATICHE

ABS(X)	Valore assoluto di X
EXP(X)	Esponenziale di X
INT(X)	Parte intera di X
LOG(X)	Logaritmo neperiano di X
SGN(X)	Segno di X
SQR(X)	Radice quadrata di X

2.2.16 Organizzazione della memoria nel Basic Applesoft

• Introduzione

Questo paragrafo descrive la scheda di memoria dell'Applesoft, l'organizzazione delle variabili, delle istruzioni, lo spazio occupato. Anche se non è vostra intenzione lavorare in assembler, la lettura delle descrizioni che seguono vi può essere utile per ridurre le dimensioni di un programma e la

sua velocità di esecuzione, oppure per comprendere il programma PHONE LIST residente sul dischetto System Master del DOS 3.3.

• Scheda di memoria

La configurazione della memoria descritta nella tabella seguente, corrisponde a quella del Basic Applesoft residente in memoria ROM.

Indirizzi	Significato
da $F7FF a $D000	Applesoft
HIMEM	Stringa di caratteri
	Spazio libero
	Tabelle di numeri e di caratteri
LOMEM	Variabili semplici
	Programma Basic
$B01	
da $FF a $00	Variabili di sistema (Applesoft, Monitor, DOS 3.3)

• Utilizzo della pagina di memoria zero

Questa zona di memoria è limitata dagli indirizzi $0000 e $00FF e contiene le variabili di sistema del Basic, del Monitor e del DOS 3.3. Descriveremo qui sotto gli indirizzi di memoria utilizzati dall'Applesoft in pagina zero, il che è molto importante da conoscere.

Indirizzi decimali	Indirizzi esadecimali	Utilizzo
00-05	$00-$05	Questi indirizzi sono utilizzati al momento dell'avviamento del sistema per determinare la presenza o no di un Monitor e per caricare il DOS necessario
10-12	$0A-$0C	Istruzione di salto alla funzione USR (cfr. par. 2.2.7)
13-25	$0D-$17	Variabili interne dell'Applesoft
32-79	$20-$4F	Indirizzi utilizzati dal Monitor (cfr. par. 4.3.4)

(segue)

(seguito)

Indirizzi decimali	Indirizzi esadecimali	Utilizzo
98-102	\$62-\$66	Risultato dell'ultima moltiplicazione/divisione eseguita
103-104	\$67-\$68	Indirizzo di inizio del programma Basic in memoria
105-106	\$69-\$6A	Indirizzo di partenza delle variabili semplici in memoria LOMEM
107-108	\$6B-\$6C	Indirizzo di partenza delle tabelle in memoria
109-110	\$6D-\$6E	Indirizzo finale delle tabelle in memoria (indirizzo di partenza-1 della zona libera)
111-112	\$6F-\$70	Indirizzo di partenza delle stringhe di caratteri in memoria. Queste sono immagazzinate tra quest'indirizzo e HIMEM
113-114	\$71-\$72	Puntatore di utilizzo generico
115-116	\$73-\$74	HIMEM
117-118	\$75-\$76	Numero di riga dell'istruzione corrente
119-120	\$77-\$78	Numero di riga nella quale si è riscontrata una delle seguenti istruzioni: CTRL-C, STOP, END
121-122	\$79-\$7A	Indirizzo della prossima istruzione da eseguire
123-124	\$7B-\$7C	Numero di riga dell'istruzione DATA trattata
125-126	\$7D-\$7E	Indirizzo del prossimo dato da leggere nelle istruzioni DATA
127-128	\$7E-\$80	Indirizzo dei dati letti: - per un'istruzione INPUT, il codice predefinito è \$201 - per un'istruzione READ, questo puntatore contiene l'indirizzo del prossimo dato dell'istruzione DATA in corso di esecuzione
129-130	\$81-\$82	Nome dell'ultima variabile trattata
131-132	\$83-\$84	Indirizzo contenente il valore dell'ultima variabile esaminata

(segue)

(seguito)

Indirizzi decimali	Indirizzi esadecimali	Utilizzo
157-163	$9D-$A3	Accumulatore flottante principale (zona di memoria contenente i numeri reali da trattare e i risultati dei calcoli fra i numeri reali)
165-171	$A5-$AB	Secondo accumulatore flottante
175-176	$AF-$B0	Indirizzo finale del programma
177-200	$B1-$C8	Sottoprogramma di acquisizione di un carattere CHRGET
184-185	$B8-$B9	Indirizzo dell'ultimo carattere acquisito da CHRGET
201-205	$C9-$CD	Numero casuale generato
216-223	$D8-$DF	Trattamento degli errori
216	$D8	Indicatore di attivazione del trattamento degli errori
218-219	$DA-$DB	Numero di riga alla quale si è verificato l'errore
222	$DE	Codice di errore (cfr. par. 2.2.12)
224-226	$E0-$E2	Coordinate X, Y di un punto grafico ad alta risoluzione
228	$E4	Colore ad alta risoluzione
232-233	$E8-$E9	Indirizzo di partenza della tabella delle figure grafiche (cfr. 3.3 e 3.4)

• Organizzazione delle variabili in memoria

Variabili semplici: una variabile semplice è rappresentata su 7 bytes. I due primi bytes contengono il nome della variabile e il suo tipo. Questo è codificato sul bit di maggiore peso dei due bytes, nel seguente modo:

reale:
- primo byte positivo
- secondo byte positivo

intero:
- primo byte negativo
- secondo byte negativo

stringa:
- primo byte negativo
- secondo byte positivo

I cinque ultimi bytes hanno un formato diverso a seconda del tipo della variabile.

VARIABILE INTERA

Numero di byte	Significato	Codifica
0	Nome	Bits 0–6: codice ASCII del primo carattere 7: tipo della variabile (vedere sopra)
1		Idem per il secondo carattere
2	Valore numerico	Byte di peso maggiore
3		Byte di peso minore
4	0	
5	0	
6	0	

VARIABILI REALI

Numero di byte	Significato	Codifica
0		Idem con un tipo diverso
1		
2	Esponente	Byte segnato (bit di segno = 7)
3	Mantissa	Byte segnato di peso maggiore
4	Mantissa	
5	Mantissa	
6	Mantissa	Byte di peso minore

La lunghezza delle stringhe di caratteri è, al massimo, di 255 caratteri; la zona di memoria sotto HIMEM contiene i valori delle stringhe.

STRINGHE DI CARATTERI

Numero di byte	Significato	Codifica
0		Idem con una codifica differente
1		
2	Lunghezza stringa	
3	Indirizzo stringa	Peso maggiore
4		Peso minore
5	0	
6	0	

TABELLE

I primi due bytes hanno lo stesso formato di quello delle variabili semplici (nome e codifica di tipo). Viene in seguito indicata la distanza (displacement) che esiste fra l'indirizzo della variabile corrente e quello di quella successiva. Vengono quindi inserite delle informazioni relative al numero di dimensioni, alla loro misura, secondo il seguente formato:

Numero di byte	Significato
3	Numero N di dimensioni
4	Byte di peso maggiore contenente la misura della dimensione 1
5	Byte di peso minore
...	
...	
4+(n—1)*2	Byte di peso maggiore contenente la misura della dimensione N
5+(n—1)*2	Byte di peso minore

A seconda del tipo di variabile, i seguenti bytes sono così ripetuti:

- reale: 5 bytes aventi lo stesso significato dei bytes da 3 a 6 della variabile semplice di tipo reale (v. sopra);
- intero: bytes da 3 a 4 di una variabile semplice intera (valore numerico);
- stringa: bytes 3, 4 e 5 di una variabile di tipo stringa di caratteri semplici (lunghezza indirizzo).

Consigli per ridurre la durata di un programma: durante l'esecuzione di un programma, il Basic riempie la zona di memoria destinata alle variabili a mano a mano che esse vengono incontrate. D'altra parte l'Applesoft, onde poter lavorare su una variabile sia essa semplice o di tipo tabella, percorre la zona di memoria corrispondente alle variabili finché trova la variabile desiderata. Solo allora ne scrive l'indirizzo nelle celle di memoria $81-$82.
Per migliorare la velocità di un programma, dovrete utilizzare le variabili incontrate per prime nel programma come variabili di lavoro.
Il fatto di utilizzare delle variabili in luogo di costanti, consente la riduzione del tempo di esecuzione di un programma di un fattore che può raggiungere 10.

Esempio: se utilizzate il numero Pi=3.14159, non ripetetelo durante il vostro programma; create piuttosto la variabile Pi contenente la quantità 3.14159.

• Organizzazione delle istruzioni

L'Applesoft codifica le parole chiave riconosciute su un byte. Le altre parti delle istruzioni sono rappresentate mediante il loro codice ASCII.
La codifica delle parole chiave avviene secondo le modalità indicate nella tabella seguente.
Per raggiungere un'istruzione, o una riga, l'Applesoft percorre sequenzialmente il programma finché essa viene incontrata. Vi consigliamo, per

Tab. 1

Codice	Parola chiave	Codice	Parola chiave	Codice	Parola chiave
128	END	138	PR#	148	DRAW
129	FOR	139	IN#	149	XDRAW
130	NEXT	140	CALL	150	HTAB
131	DATA	141	PLOT	151	HOME
132	INPUT	142	HLIN	152	ROT=
133	DEL	143	VLIN	153	SCALE=
134	DIM	144	HGR2	154	SHLOD
135	READ	145	HGR	155	TRACE
136	GR	146	HCOLOR=	156	NOTRACE
137	TEXT	147	HPLOT	157	NORMAL
158	INVERSE	184	DEF	210	SGN
159	FLASH	185	POKE	211	INT
160	COLOR=	186	PRINT	212	ABS
161	POP	187	CONT	213	USR
162	VTAB	188	LIST	214	FRE
163	HIMEM:	189	CLEAR	215	SCRN(
164	LOMEM:	190	GET	216	PDL
165	ONERR	191	NEW	217	POS
166	RESUME	192	TAB(	218	SQR
167	RECALL	193	TO	219	RND
168	STORE	194	FN	220	LOG
169	SPEED=	195	SPC(	221	EXP
170	LET	196	THEN	222	COS
171	GOTO	197	AT	223	SIN
172	RUN	198	NOT	224	TAN
173	IF	199	STEP	225	ATN
174	RESTORE	200	+	226	PEEK
175	&	201	—	227	LEN
176	GOSUB	202	*	228	STR$
177	RETURN	203	/	229	VAL
178	REM	204	∧	230	ASC
179	STOP	205	AND	231	CHR$
180	ON	206	OR	232	LEFT$
181	WAIT	207	>	233	RIGHT$
182	LOAD	208	=	234	MID$
183	SAVE	209	<		

incrementare la velocità dei vostri programmi, ma non la loro leggibilità, di inserire tutti i vostri sottoprogrammi in testa secondo il seguente formato:

GOTO programma principale
Sottoprogrammi
...
Programma principale

Per una miglior comprensione dei programmi che riporteremo qui di seguito, questo formato non verrà da noi più adoperato negli esempi di questo testo.
Eliminare le istruzioni REM riduce la misura e la durata dei vostri programmi. Ma non eseguite tale operazione prima di aver terminato la fase di messa a punto del vostro programma.

• **Modifica dell'indirizzo di caricamento di un programma Basic**

Per fare ciò, è sufficiente posizionare il puntatore posto all'indirizzo 103-104 al suo nuovo indirizzo di partenza. Le successive operazioni sono uguali alle precedenti. ATTENZIONE a non sporcare la memoria, ai valori di LOMEM e di HIMEM.

2.3 UTILIZZO DI UNA STAMPANTE

2.3.1 Introduzione

L'Apple II può essere munito di una stampante. Per ottenere ciò, è necessario possedere una scheda di espansione. Questa può essere di due tipi:

- Interfaccia parallela (gli 8 bits di un byte sono emessi in parallelo su 8 fili), inserita nel connettore numero 1.
- Interfaccia seriale asincrona (gli 8 bits sono emessi uno dietro l'altro sullo stesso filo), inserita nel connettore numero 2.

La scelta di una stampante deve essere fatta secondo il tipo di connessione e secondo la qualità desiderata. Esistono a questo scopo due tipi di stampanti:

- Stampante a margherita, di qualità professionale, necessaria per eseguire il trattamento di testi.
- Stampante ad aghetti, di qualità inferiore per le lettere, ma che consente possibilità grafiche e realizza delle copie dello schermo.

2.3.2 Presentazione dell'EPSON MX82FT

L'EPSON MX82FT può essere collegata all'Apple II sia tramite una scheda di interfaccia parallela, sia mediante una scheda di espansione appositamen-

te concepita dall'Epson per facilitare la gestione delle stampe (copia di schermo, numero di caratteri per riga, stampa di lettere minuscole, caratteri in grassetto, funzione video inversa...).
Il trascinamento può essere fatto tramite trazione o frizione. La velocità di stampa è di 80 caratteri al secondo.

1. *Le caratteristiche sopra indicate sono relative al tipo* III MX82FT.
2. *Supporremo nel seguito di questo paragrafo che la scheda di interfaccia sia stata inserita nel connettore numero 1.*

• Stampa di testi

Caratteri speciali: le diverse scelte possibili sono effettuate inviando dei codici speciali oppure stampando dei valori numerici nelle celle di memoria.

- Numero di caratteri per riga POKE 1657,n
- Minuscole CRTL-W viene utilizzato da delimitatore PRINT"A" + CHR$(23) + "B" + CHR$(23) + "C" fornisce AbC.

È possibile cambiare il delimitatore delle minuscole inserendo in nuovo codice ASCII *all'indirizzo* 1401.

- Doppia stampa { inizio POKE 1529,255 / fine POKE 1529,0

Le due stampe delle righe sono spaziate in senso verticale da 1/256 *di pollice.*

- Caratteri condensati { inizio PRINT CHR$(15) o invio del codice SI(CTRL-O) / fine PRINT CHR$(18) o invio del codice DC2(CTRL-R)
- Caratteri ingranditi { inizio PRINT CHR$(14) o invio del codice SO(CTRL-N) / fine PRINT CHR$(20) o invio del codice DC4(CTRL-T)

Per questi ultimi due tipi di caratteri ogni fine di riga provoca il loro rilascio.

- Caratteri in grassetto { inizio PRINT CHR$(27) + "E" o invio dei codici ESC E / fine PRINT CHR$(27) + "F" o invio dei codici ESC F
- Sottolineatura inizio e fine PRINT CHR$(27) + "—" o invio codici ESC —.

Scelta del campo di caratteri: i campi di caratteri esistono in:

- inglese (0);
- francese (1);
- tedesco (2);
- ecc.

Questo consente in particolare, di poter disporre di un campo di caratteri dotati di accento.

Modifica della spaziatura delle righe:

- 1/8 di pollice = ESC 0
- 7/72 di pollice = ESC 1
- 1/6 di pollice = ESC 2
- n/216 di pollice = ESC 3n

• **Grafica ad alta risoluzione**

La scheda di espansione contiene un programma di copia dello schermo. Per utilizzarlo è sufficiente eseguire il comando CTRL-Q quando la stampante è in funzione. La copia ha luogo secondo il valore del byte posto all'indirizzo 1913.

N	D	I	E	O	A	S	P

Il ruolo dei diversi bits è il seguente:

N = 0 { Immagine completa
128 { Riga corrente del cursore

D = 0 { Misura dell'immagine standard
64 { Misura dell'immagine doppia

I = 0 { Funzione normale
32 { Funzione inversa

E,A,O Operazioni bit a bit tra le due pagine grafiche
{ E = 16 Or esclusivo
A = 8 And (prodotto logico)
O = 4 Or (somma logica)

S = 0 2 { Non viene stampata la pagina 2
Stampa della pagina 2

P = 0 1 { Non viene stampata la pagina 1
Stampa della pagina 1

È possibile ottenere una migliore risoluzione grafica eseguendo

POKE 1145,DD oppure DD = 76 per una migliore risoluzione
DD = 75 per una risoluzione normale

In più, potete programmare direttamente le immagini bit a bit. Gli esempi rappresentati nelle figure da 3.3 a 3.6 sono stati ottenuti con un programma di copia dello schermo.

2.4 ESEMPI DI PRELEVAMENTO DEI DATI

2.4.1 Introduzione

Ecco degli esempi classici di sottoprogrammi Basic per un'applicazione gestionale di un'agenda di indirizzi, su un file ad accesso diretto.

Qui di seguito abbiamo esemplificato un Menu, un esempio di prelevamento dati, un esempio di stampa di dati prelevati, di modifica di dati ed infine un ultimo esempio di ricerca dati nel file così costituito.
Abbiamo soppresso la gestione del file precisando però dove essa ha luogo.

2.4.2 Menu

```
430  REM:
440  REM   VISUALIZZAZIONE DEL MENU
450  REM  ******************
460  REM
470  HOME : INVERSE
480  PRINT "     *** AGENDA INDIRIZZI  ***": PRINT : PRINT : PRINT
490  PRINT "1- VISUALIZZAZIONE CON/SENZA MODIFICHE"
500  PRINT "2- INSERIMENTO (NUOVO INDIRIZZO ) "
510  PRINT "3- LISTA DEL FILE  "
520  PRINT "4- FINE PROGRAMMA  "
530  PRINT
540  INPUT "SCEGLI UN NUMERO  (1,2,3,4) ";A
550  IF A < 1 OR A > 4 THEN 470
560  NORMAL
570  ON A GOSUB 640,1490,1960,2180
580  GOTO 470
```

2.4.3 Prelevamento dati

I dati da prelevare sono i seguenti:

- Cognome	(COGN$)
- Nome	(NOM$)
- Tipo di via	(TIPO$)
- Nome della via	(NVIA$)
- Numero	(NUMERO$)
- Città	(CITTA$)
- Codice postale	(CDPOST$)
- Prefisso	(INDTEL$)
- Numero di telefono	(TEL$)

I dati possono essere introdotti all'atto di un nuovo inserimento oppure quando sopraggiunge una modifica; un sottoprogramma di prelevamento dati è stato realizzato per ciascuna di queste funzioni.

Inserimento di una nuova registrazione

```
1470  REM
1480  REM  ************************************
1490  REM      AGGIUNTA DATI  (2)
1500  REM  ************************************
1510  REM
1520  INVERSE
1530  HOME : PRINT "NUOVO INDIRIZZO ": PRINT : PRINT
1540  NORMAL
1550  PRINT "SE UN DATO NON PUO' ESSERE INSERITO"
1560  PRINT "PREMERE SOLO  RETURN  "
1570  PRINT
1580 CODE$ = "1"
1590  GOSUB 3000: REM    SCELTA DEL COGNOME
1595 B$ = COGN$
1600  GOSUB 3100: REM   SCELTE DEL NOME
1610  GOSUB 3200: REM   SCELTA DEL TIPO DI VIA
1620  GOSUB 3300: REM   SCELTA DEL NOME DELLA VIA
1630  GOSUB 3400: REM   SCELTA DEL NUMERO
1640  GOSUB 3500: REM   SCELTA DELLA CITTA'
1650  GOSUB 3600: REM   SCELTA DEL CODICE POSTALE
1660  GOSUB 3700: REM   SCELTA DEL TELEFONO
1670  HOME
1680  INVERSE
1690  PRINT "INFORMAZIONI SCELTE "
1700  NORMAL : PRINT : PRINT
1710  GOSUB 4000: REM   AGGIUNTA NEL BUFFER
1720  PRINT : INVERSE
1730  INPUT "CONVALIDI QUESTE INFORMAZIONI (Y/N)";A$
1740  IF A$ <  > "N" AND A$ <  > "Y" THEN 1730
1745  IF A$ = "N" THEN 1490
1750  REM
1760  REM   AGGIUNTA DEL RECORD NEL FILE
1770  REM
```

scrittura nel file

Sottoprogrammi di prelevamento dati:

```
3000  REM
3010  REM   SCELTA DEL COGNOME
3020  REM  *************
3030  REM
3040  INPUT "COGNOME ";A$
3050 N = 15: IF  LEN (A$) <  = N THEN 3080
3060  GOSUB 3920: REM   MESSAGGIO DI ERRORE
```

```
3070  GOTO 3040
3080  GOSUB 5000:COGN$ = A$
3090  RETURN
3100  REM
3110  REM   SCELTA DEL NOME
3120  REM  ****************
3130  REM
3140  INPUT " NOME  ";A$
3150 N = 15: IF  LEN (A$) <  = N THEN 3180
3160  GOSUB 3920: REM   MESSAGGIO DI ERRORE
3170  GOTO 3140
3180  GOSUB 5000:NOM$ = A$
3190  RETURN
3200  REM
3210  REM   SCELTA DEL TIPO DI STRADA
3220  REM  **********************
3230  REM
3240  INPUT "TIPO DI STRADA (VIA,CORSO...) ";A$
3250 N = 3: IF  LEN (A$) <  = N THEN 3280
3260  GOSUB 3920: REM   MESSAGGIO DI ERRORE
3270  GOTO 3240
3280  GOSUB 5000:TIPO$ = A$
3290  RETURN
3300  REM
3310  REM   SCELTA DEL NOME DELLA VIA
3320  REM  **********************
3330  REM
3340  INPUT "NOME DELLA VIA ";A$
3350 N = 15: IF  LEN (A$) <  = N THEN 3380
3360  GOSUB 3920: REM   MESSAGGIO DI ERRORE
3370  GOTO 3340
3380  GOSUB 5000:NVIA$ = A$
3390  RETURN
3400  REM
3410  REM   SCELTA DEL NUMERO
3420  REM  ****************
3430  REM
3440  INPUT "NUMERO ";A$
3450 N = 3: GOSUB 5000:NUMERO$ = A$
3460  RETURN
3470  REM
3480  REM   SCELTA DELLA CITTA'
3490  REM  ******************
3500  REM
3510  INPUT "CITTA' ";A$
3520 N = 15: IF  LEN (A$) <  = N THEN 3550
3530  GOSUB 3870: REM   MESSAGGIO DI ERRORE
3540  GOTO 3510
3550  GOSUB 5000:CITTA$ = A$
3560  RETURN
3570  REM
```

```
3580  REM   SCELTA DEL CODICE POSTALE
3590  REM  ********************
3600  REM
3610  INPUT "CODICE POSTALE";A$
3620 N = 5: IF  LEN (A$) <  = N THEN 3650
3630  PRINT "NON PIU' DI CINQUE CIFRE "
3640  GOTO 3610
3650  GOSUB 5000:CDPST$ = A$
3660  RETURN
3670  REM
3680  REM    SCELTA DEL PREFISSO
3690  REM  **********************************
3700  REM
3710  INPUT "PREFISSO TELEFONICO ";A$
3720 N = 2: IF  LEN (A$) <  = N THEN 3750
3730  PRINT "NON PIU' DI DUE CIFRE"
3740  GOTO 3710
3750 N = 2: GOSUB 5000:INDTEL$ = A$
3760  REM
3770  REM   SCELTA DEL NUMERO DI TELEFONO
3780  REM  *****************************
3790  REM
3800  INPUT "NUMERO DI TELEFONO ";A$
3810 N = 7: IF  LEN (A$) <  = N THEN 3840
3820  PRINT "NON PIU' DI SETTE CIFRE"
3830  GOTO 3800
3840  GOSUB 5000:TEL$ = A$
3850  RETURN
5000  REM
5010  REM   QUADRO DEI DATI SCELTI
5020  REM  *****************************
5030  REM
5040  REM   IN UNA CATENA DI N CARATTERI
5050  REM  *****************************
5055 N = N -  LEN (A$)
5056  IF N <  = 0 THEN  RETURN
5060 A$ = A$ +  LEFT$ (C$,N)
5090  RETURN
```

Questi sottoprogrammi richiamano quello posto in riga 3960 per la stampa del messaggio di errore "STRINGA TROPPO LUNGA"

```
3860  REM
3870  REM
3880  REM   VISUALIZZAZIONE DEI MESSAGGI D'ERRORE
3890  REM   STRINGA TROPPO LUNGA
3900  REM  **************************
```

```
3910  REM
3920  PRINT : INVERSE
3930  PRINT "NON PIU' DI ";N;" CARATTERI,PREGO"
3940  NORMAL
3950  RETURN
```

2.4.4 Visualizzazione dei dati

Sullo schermo:

```
3960  REM
3970  REM   VISUALIZZAZIONE DEI DATI SCELTI
3980  REM  *****************************
3990  REM
4000  PRINT
4010  PRINT "COGNOME"; TAB( 20);COGN$
4020  PRINT "NOME"; TAB( 20);NOM$
4030  PRINT "INDIRIZZO"; TAB( 20);NUMERO$;" ";TIPO$;" ";NVIA$
4040  PRINT "CITTA'"; TAB( 20);CDPST$;" ";CITTA$
4050  PRINT "TELEFONO "; TAB( 20);"(";INDTEL$;")";TEL$
4060  RETURN
```

Su stampante:

```
3960  REM
3970  REM   STAMPA DEI DATI SCELTI
3980  REM  *****************************
3990  REM
4000  PR#1
4010  PRINT "COGNOME"; TAB( 20);COGN$
4020  PRINT "NOME"; TAB( 20);NOM$
4030  PRINT "INDIRIZZO"; TAB( 20);NUMERO$;" ";TIPO$;" ";NVIA$
4040  PRINT "CITTA'"; TAB( 20);CDPST$;" ";CITTA$
4050  PRINT "TELEFONO "; TAB( 20);"(";INDTEL$;")";TEL$
4060  RETURN
```

2.4.5 Ricerca di un dato

Si possono riscontrare tre fasi successive:

1. Determinazione del modo di procedere.
2. Ricerca nel file.

3. Test di concordanza.
4. Se il test è negativo, riprendere dalla fase 2 finché non si incontra la fine del flusso.

```
650  REM  ***********************************
660  REM   RICERCA DENTRO IL FILE   (1)
670  REM  ***********************************
680  REM
690  HOME : INVERSE : PRINT "   *** VISUALIZZAZIONE *": NORMAL
700  PRINT : PRINT : PRINT
710  PRINT "DA DOVE INIZIA LA RICERCA        "
720  PRINT
730  PRINT "1-IL COGNOME"
740  PRINT "2-IL NOME"
750  PRINT "3-L'INDIRIZZO"
760  PRINT "4-LA CITTA'"
770  PRINT "5-IL NUMERO DI CODICE POSTALE"
780  PRINT "6-IL NUMERO DI TELEFONO "
790  PRINT : PRINT : INPUT "SCEGLIERE  (INSERIRE ANCHE PIU' CIFRE )
";B
810 RECH$(1,2) = "RICERCA SUL COGNOME"
820 RECH$(2,2) = "RICERCA SUL NOME "
830 RECH$(3,2) = "RICERCA SUL INDIRIZZO"
840 RECH$(4,2) = "RICERCA SULLA CITTA'"
850 RECH$(5,2) = "RICERCA SUL CODICE POSTRALE"
860 RECH$(6,2) = "RICERCA SUL NUMERO DI TELEFONO "
870 RECH(1,2) = 15:RECH(2,2) = 15:RECH(3,2) = 15:RECH(4,2) =
15:RECH(5,2) = 5:RECH(6,2) = 7
930 B =  LEN (B$)
940  FOR I = 1 TO 6
950 RECH(I,1) = 0
960  FOR J = 1 TO B
970  IF  VAL ( MID$ (B$,J,1)) <  > I THEN 1010
980 RECH(I,1) = 1
990  PRINT RECH$(I,2): INPUT " ";A$
1000 N = RECH(I,2): GOSUB 5000:RECH$(I,1) = A$
1010  NEXT J
1020  NEXT I
1030  REM
1040  REM   FASE DI RICERCA
1050  REM  ************************
1060  REM
1100  IF NB > 1 THEN 1120
1110  PRINT "FILE VUOTO  ": INPUT "PREMERE RETURN PER CONTINUARE";A:
RETURN
1120  FOR I = 1 TO NB - 1
1130 OK = 1
1140  PRINT RD$ +  STR$ (I)
```

```
1150  GOSUB 2910
1160  PRINT D$
1170  REM
1180  REM   TEST DI UGUAGLIANZA
1190  REM  ******************
1200  REM
1210  REM   ARTICOLI -DATI DA RICERCARE
1220  REM  ******************************
1230  REM
1235 OK = 1
1240  IF RECH(1,1) = 0 THEN 1260
1250 OK = OK AND (COGN$ = RECH$(1,1))
1260  IF RECH(2,1) = 0 THEN 1280
1270 OK = OK AND (NOM$ = RECH$(2,1))
1280  IF RECH(3,1) = 0 THEN 1310
1290 A$ = NUMERO$ + " " + TIPO$ + " " + NVIA$
1300 OK = OK AND A$ = RECH$(3,1))
1310  IF RECH(4,1) = 0 THEN 1330
1320 OK = OK AND (CITTA$ = RECH$(4,1))
1330  IF RECH(5,1) = 0 THEN 1350
1340 OK = OK AND (CDPST$ = RECH$(5,1))
1350  IF RECH(6,1) = 0 THEN 1370
1360 OK = OK AND (TEL$ = RECH$(6,1))
1370  IF OK = 0 THEN 1440
1375  PRINT "1375"
1380  HOME : INVERSE : PRINT "DATI POSSIBILI ": NORMAL
1390  PRINT : PRINT : PRINT : GOSUB 4000: PRINT
1400  PRINT "VUOI MODIFICARE O CANCELLARE "
1410  INPUT "QUESTA REGISTRAZIONE (Y/N)";A$
1420  IF A$ <  > "N" AND A$ <  > "Y" THEN 1400
1430  IF A$ = "Y" THEN  GOSUB 2310
1440  NEXT I
1450  RETURN
```

2.4.6 Modifica/Eliminazione di un elemento

```
2320  REM   MODIFICA / CANCELLAZIONE
2330  REM  ************************
2340  REM
2350  REM   DI UNA REGISTRAZIONE
2360  REM  ******************
2370  REM
2380  PRINT : PRINT "VUOI"
2390  PRINT "MODIFICARE IL RECORD (1)"
2400  PRINT "CANCELLARE IL RECORD (2)"
2410  INPUT "SCEGLI ";A
2420  IF A < 1 OR A > 2 THEN 2380
```

```
2430  ON A GOTO 2440,2700
2440  REM
2450  REM   MODIFICA DI UN RECORD
2460  REM  *******************
2470  REM
2480  PRINT : PRINT "CHE COSA VUOI MODIFICARE"
2490  PRINT : PRINT "1-IL COGNOME": PRINT "2-IL NOME  ": PRINT
"3-L'INDIRIZZO": PRINT "4-LA CITTA'": PRINT "5-IL CODICE POSTALE":
PRINT "6-IL NUMERO DI TELEFONO"
2500  PRINT : INPUT "SCEGLI UN NUMERO (1,2,3,4,5,6) ";A
2510  IF A <  > 3 THEN 2560
2520  GOSUB 3200: REM   TIPO DI VIA
2530  GOSUB 3300: REM   NOME DELLA VIA
2540  GOSUB 3400: REM   NUMERO
2550  GOTO 2570
2560  ON A GOSUB 3000,3100,,3470,3570,3670
2570  HOME : INVERSE : PRINT "INFORMAZIONI SCELTE ": NORMAL
2580  GOSUB 4060

2590  PRINT : PRINT "CHE SCEGLI"
2600  PRINT "1-MODIFICARE UN ALTRO DATO DEL RECORD"
2610  PRINT "2-INSERIRE LE MODIFICHE"
2620  PRINT "3-NON INSERIRE LE MODIFICHE"
2630  INPUT "SCEGLI UN NNUMERO (1,2,3)";A
2640  ON A GOTO 2440,2650,2690
2650  REM
2660  REM   REGISTRAZIONE DELLE MODIFICHE
2670  REM
2680  PRINT WR$;I: GOSUB 2990: PRINT D$
2690  RETURN
2700  REM
2710  REM   CANCELLAZIONE DI UN RECORD
2720  REM  ************************
2730  REM
```

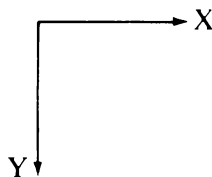
CAPITOLO 3

Possibilità grafiche e sonore

3.1 INTRODUZIONE

Il vostro sistema Apple II offre numerose possibilità grafiche e sonore, che aggiungono un grande interesse ai programmi di cui disponete. Potete creare i più diversi tipi di giochi come gli scacchi, il Bridge, Guerre Stellari, Simulazione del pilotaggio di un'aereonave, ecc...
Le due funzioni grafiche seguenti sono disponibili sull'Apple II:

- Funzione grafica a bassa risoluzione in cui lo schermo è diviso in 48 righe di 40 colonne ciascuna e in cui ogni punto appare sotto forma di un rettangolo sullo schermo (mezzo carattere).
- Funzione grafica ad alta risoluzione in cui la definizione è di 280 punti (da 0 a 279) in senso orizzontale e di 192 punti (da 0 a 191) in senso verticale.

Con una scheda a 80 colonne più 64 K di espansione di memoria, la risoluzione orizzontale è portata a 560 punti in senso orizzontale e in bianco e nero.
In queste due funzioni grafiche, l'asse delle ascisse X è posto nella parte superiore dello schermo, mentre l'asse delle ordinate Y si trova nella parte sinistra dello schermo e diretto verso il basso. Questa disposizione è comune a tutti i Basic realizzati dalla Microsoft.
Inoltre l'Applesoft vi permette di creare delle figure grafiche ad alta risoluzione precedentemente memorizzate che in seguito potrete manipolare facilmente.

3.2 LE FUNZIONI GRAFICHE A BASSA RISOLUZIONE

3.2.1 Come innescare la funzione

Per passare dalla funzione "testo" alla funzione grafica a bassa risoluzione, l'istruzione GR cancella lo schermo e lo colloca nella funzione mista "testografica" (40 righe grafiche e 4 righe di testo nella parte inferiore dello schermo).
Infine pone il cursore all'inizio del testo.
Potete eliminare queste 4 righe di testo mediante l'istruzione

POKE-16302,0

Lo schermo così ottenuto comprende 48 righe grafiche.
Potrete rigenerare la finestra delle 4 righe della parte inferiore dello schermo tramite l'istruzione

POKE-16301,0

Per ritornare alla funzione "testo", vi sarà sufficiente utilizzare l'istruzione TEXT.
Sulle prime 20 (oppure 24) righe dello schermo appariranno allora degli arabeschi.
Questo è assolutamente normale, poiché l'Apple II utilizza la stessa zona di memoria sia per la funzione "testo" sia per la funzione grafica a bassa risoluzione.

3.2.2 Istruzioni grafiche

I due Basic dell'Apple II consentono le seguenti istruzioni grafiche a bassa risoluzione:

COLOR = espressione

Scelta del colore con il quale saranno eseguite le successive visualizzazioni fra i 16 colori seguenti:

0. nero
1. magenta
2. blu scuro
3. porpora
4. verde scuro
5. grigio
6. blu
7. azzurro
8. marrone
9. arancione
10. grigio
11. rosa

12. verde
13. giallo
14. trasparente
15. bianco

Il valore dell'espressione deve essere compreso fra 0 e 255 e va considerato in modulo 16. Ad esempio l'istruzione COLOR = 40 selezionerà il colore azzurro.
Il colore attivato è riportato a 0 *mediante l'istruzione* GR.

PLOT espressione X, espressione Y

+ → X Visualizzazione di un punto all'intersezione della colonna X = 39 e della riga Y = 47

↓ Y

Se le coordinate non sono comprese negli intervalli indicati, il Basic visualizzerà il seguente messaggio di errore:

ILLEGAL QUANTITY ERROR

e interromperà l'esecuzione del programma.
Ecco un esempio che consente il tracciamento di una riga diagonale sullo schermo

GR:COLOR = 12:FOR X = 0 TO 39:PLOT X,39—X:NEXT X

HLIN X1,X2 AT Y

Tracciamento di una riga orizzontale sullo schermo tra i punti (X1,Y) e (X2,Y).

VLIN Y1,Y2 AT X

Tracciamento di una riga verticale tra i punti (X,Y1) e (X,Y2).
L'esempio seguente vi indica come tracciare un quadrato.

```
10 REM DISEGNO DI UN QUADRATO
20 GR
30 COLOR=12
40 HLIN 20,30 AT 20
50 VLIN 20,30 AT 20
60 HLIN 20,30 AT 30
70 VLIN 20,30 AT 30
80 END
```

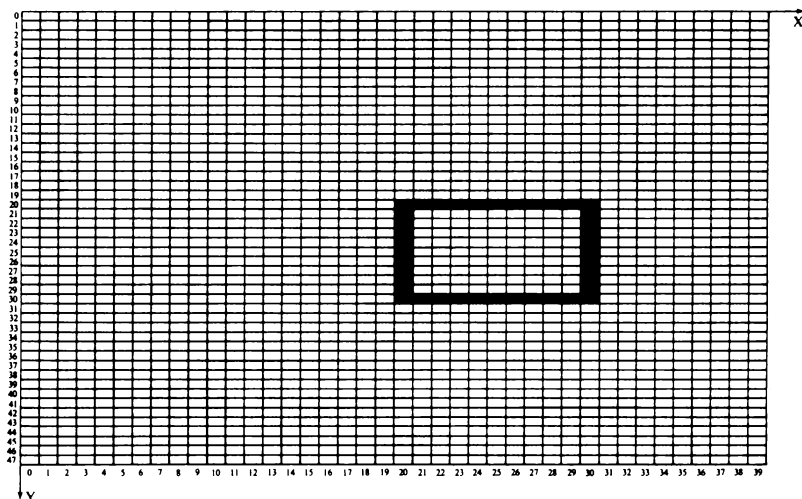



Fig. 3.1. Tracciamento di un quadrato di coordinate uguali (ascisse e ordinate).

Noterete che i lati del quadrato non hanno gli stessi spessori (cfr. fig. 3.1). Questo è dovuto al fatto che, nella funzione grafica a bassa risoluzione, un punto è un carattere tagliato in due. Ne risulta che esso costituisce un rettangolo i cui lati orizzontali sono in rapporto di 2 a 1 rispetto ai verticali.

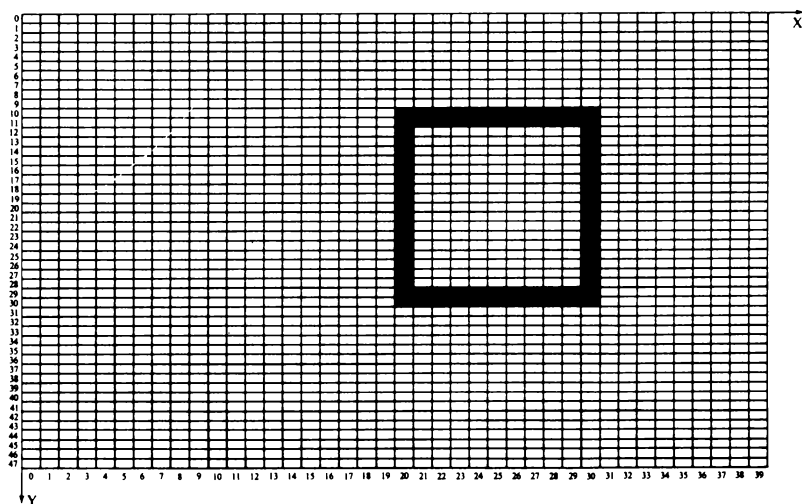



Fig. 3.2. Tracciamento di un quadrato vero e proprio nella funzione a bassa risoluzione.

Il programma seguente consente il disegno di un quadrato vero e proprio sullo schermo

```
10 REM DISEGNO DI UN QUADRATO
20 GR
30 COLOR=12
40 HLIN 20,30 AT 10
50 HLIN 20,30 AT 11
60 HLIN 20,30 AT 29
70 HLIN 20,30 AT 30
80 VLIN 10,30 AT 20
90 VLIN 10,30 AT 30
100 END
```

Provate! Il quadrato ottenuto è più regolare (cfr. fig. 3.2).

Funzione SCRN(X,Y)

Visualizzazione del numero del colore del punto di coordinate (X,Y). Sostituite la riga 100 con la seguente:

100 COLOR = SCRN(20,30) + 1:FOR I = 1 TO 1000:
NEXT I:GOTO 40

Provate ad eseguire il programma; vedrete allora cambiare periodicamente il colore del quadrato sullo schermo.

3.2.3 Utilizzo della memoria

L'Apple II mette a disposizione due pagine (zone di memoria) grafiche a bassa risoluzione. La prima pagina è quella utilizzata dal Basic e dal programma Monitor. La seconda non è gestita né dal Monitor, né dal Basic. Avete la possibilità, se desiderate sfruttarla, di calcolare l'indirizzo in memoria dei diversi punti. Potrete riferirvi a questo proposito al capitolo 4, riguardante il Monitor, per conoscere gli indirizzi e i sottoprogrammi a disposizione.

3.2.4 Esempio

Ecco un programma Basic che riconosce le manopole per i giochi e visualizza sullo schermo il punto di coordinate corrispondente al valore letto sulle manopole, modulo 40.

```
10 REM VISUALIZZAZIONE DEI PUNTI LE CUI COORDINATE VENGONO LETTE
20 REM TRAMITE LE JOYSTICK
30 REM
40 REM COPYRIGHT DE MERLY-
```

```
50 REM
60 DIM XY(1)
70 GR
80 VTAB 21:HTAB 1:CALL -958
90 COLOR=14
100 FOR I=0 TO 1
110 FOR J=0 TO 10:NEXT
120 XY(I)=PDL(I)
130 IF XY(I)>39 THEN XY(I)=39
140 NEXT
150 VTAB 21:PRINT "X= ";XY(0),"Y= ";XY(1)
160 PLOT XY(0),XY(1)
170 GOTO 100
```

Il funzionamento della funzione PDL *è spiegato più in dettaglio nel paragrafo* 3.7.

3.3 LA FUNZIONE GRAFICA AD ALTA RISOLUZIONE

3.3.1 Introduzione

In questa funzione grafica, la più interessante dell'Apple II, lo schermo è decomposto in punti e non più in rettangoli e, come per la funzione a bassa risoluzione, voi avete a disposizione due pagine grafiche.
Le istruzioni grafiche sono incorporate unicamente nel Basic Applesoft. Per quanto riguarda l'Integer Basic, sarà l'utilizzatore stesso a dover gestire la funzione ad alta risoluzione.
Le due pagine di memoria ad alta risoluzione non sono protette dal Basic Applesoft. Dovete dunque dare un valore alle variabili HIMEM e LOMEM. Se la lunghezza del vostro programma non supera 12 K bytes, sono possibili le due soluzioni seguenti:

- Posizionare le variabili del proprio programma utente prima delle pagine grafiche ponendo HIMEM a 8191 (16383) se desiderate lavorare sulla pagina uno (due).
- Posizionare le variabili del proprio programma utente dopo le pagine grafiche ponendo LOMEM a 16384 (24576) per utilizzare la pagina uno (due).

Se il vostro programma supera 12 K bytes, dovrete porlo dopo le pagine grafiche (cfr. par. 2.2.16).
Non potrete utilizzare sempre le pagine grafiche: questo infatti dipende dalla memoria che il vostro sistema ha a disposizione. Vi sono necessari 16 K *bytes per la pagina uno,* 24 K *bytes per la pagina due. Se desiderate disporre allo stesso tempo anche del* DOS, *dovrete aggiungere* 12 K *bytes al vostro calcolo.*

3.3.2 Come innescare la funzione

Volendo posizionare lo schermo sulla funzione grafica ad alta risoluzione, l'istruzione HGR azzera lo schermo stesso e vi pone 160 righe grafiche nella parte superiore e 4 righe di testo nella parte inferiore. Per sopprimere (reintrodurre) queste ultime, dovrete utilizzare l'istruzione:

POKE-16302,0 (POKE-16301,0)

In realtà, l'istruzione HGR *non modifica la pagina di memoria contenente il testo, ma lascia visibili solamente le* 4 *righe della parte inferiore dello schermo. Il cursore sarà visibile solamente all'interno di queste righe.*
È possibile inserirsi nella funzione grafica ad alta risoluzione senza azzerare lo schermo, utilizzando le seguenti istruzioni:

POKE-16304,0 passaggio in funzione grafica
POKE-16297,0 funzione ad alta risoluzione
POKE-16300,0 selezione della pagina uno.

Per posizionarvi sulla pagina due, dovrete utilizzare l'istruzione HGR2 invece di HGR. Per portarsi in altissima risoluzione, si impiega l'istruzione HGR3.
Se si lavora sulla pagina due ad alta risoluzione e il programma incontra l'istruzione GR, *si produce l'effetto seguente: la pagina due passa in bassa risoluzione e la pagina uno si azzera.*

3.3.3 Istruzioni grafiche ad alta risoluzione

HCOLOR = espressione

Nella funzione ad alta risoluzione esistono i seguenti codici di colore:

0. Nero 1
1. Verde
2. Viola
3. Bianco 1
4. Nero 2
5. Arancione
6. Blu
7. Bianco 2

Abbiamo volutamente evidenziato due gruppi di colori. Infatti, visualizzare il colore Bianco 1 *al punto di coordinate* (x,y), *farà apparire un punto verde se* x *è pari, viola se* x *è dispari, bianco se i due punti* (x,y) *e* (x + 1,y) *sono accesi. Il discorso è lo stesso per il secondo gruppo di colori. Questo effetto è dovuto al metodo di funzionamento dei televisori a colori.*

HPLOT x1,y1 TO x2,y2 TO...

La grafica ad alta risoluzione presenta il vantaggio di consentire il tracciamento di segmenti di retta formanti un angolo qualunque con l'orizzontale. Sono possibili diversi formati dell'istruzione:

HPLOT x,y	visualizzazione del punto di coordinate x,y
HPLOT x1,y1, TO x2,y2, (TO..)	tracciamento di segmenti tra i punti (x1,y1) e (x2,y2) e...
HPLOT TO x2,y2 (TO...)	tracciamento di un segmento tra l'ultimo punto visualizzato e il punto x2,y2.

Se in precedenza non era stato visualizzato alcun punto, viene considerata l'origine (0,0). *Le coordinate dell'ultimo numero visualizzato sono poste agli indirizzi* \$E0-\$E2 (224-226).

Esempio:

HPLOT 10,10 TO 10,20 TO 20,20 TO 20,10 TO 10,10 TO 20,20 TO 20,10 TO 10,20

Questa istruzione traccia un quadrato e le sue diagonali.

3.3.4 Esempio

Ecco un programma che disegna sullo schermo una spirale triangolare. Esso aumenta le dimensioni della spirale fino a che essa occupa completamente lo schermo, mutando il suo colore ad ogni tracciamento.

```
10 REM PROGRAMMA DI DIMOSTRAZIONE DELLE POSSIBILITA'
20 REM GRAFICHE IN ALTA RISOLUZIONE DEL BASIC APPLESOFT
30 REM
40 REM
50 REM COPYRIGHT DE MERLY
60 REM DISEGNO DI UNA SPIRALE TRIANGOLARE
70 REM
80 COLOR=5
90 XC=140:YC=96:REM CENTRO DELLO SCHERMO
100 HGR:HCOLOR=5:HPLOT XC,YC
110 LP=0:L=5:DISTANZA=5
120 REM INIZIO DEL LOOP
130 HCOLOR=COLORE
140 HPLOT TO XC+L,YC+LP
150 HPLOT TO XC,YC-L
160 HPLOT TO XC-L-DISTANZA,YC+L
170 LP=L:L=L+DISTANZA
```

```
180 COLORE=COLORE+1
190 IF COLORE=8 THEN COLORE=0
200 IF YC+L<193 THEN 130
210 END
```

Il risultato ottenuto è presentato in figura 3.3.

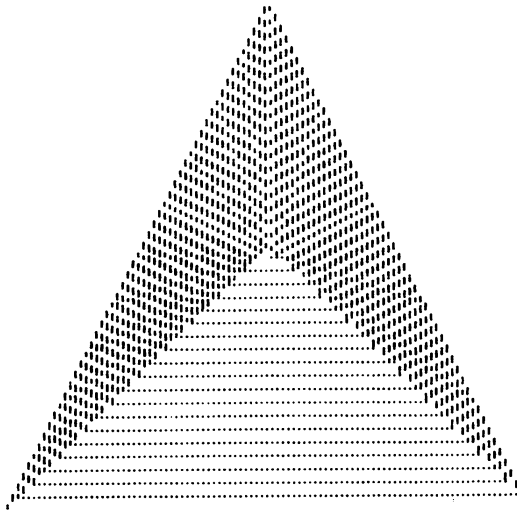



Fig. 3.3.

3.3.5 Tracciamento di curve sullo schermo

Il tracciamento di linee curve sullo schermo pone principalmente i due problemi seguenti:

1. Determinazione del fattore di scala necessario perché la curva sia contenuta nello schermo.
2. Eventuale traslazione delle coordinate per ottenere una buona quadratura.

L'esempio mostrato qui di seguito realizza il disegno di una cicloide, curva di equazioni:

$$X = K(3COS(B) + COS(3B))$$
$$Y = K(3SIN(B) - COS(3B))$$

```
10 REM DISEGNO DI UN'IPOCICLOIDE A 4 ITERAZIONI SULLO SCHERMO
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
50 HGR
60 REM DIMENSIONAMENTO DELLA FIGURA
70 SCALA=20
80 XC=140
90 YC=80
100 HCOLOR=3
110 HPLOT XC+4*SCALA,YC
120 FOR TETA=0 TO 6.28 STEP 0.02
130 X=XC+SCALA*(3*COS(TETA)+COS(3*TETA))
140 Y=YC+SCALA*(3*SIN(TETA)-SIN(3*TETA))
150 HPLOT TO X,Y
155 VTAB 21:CALL -958:PRINT "X= ";INT(X),"Y= ";INT(Y)
160 NEXT
170 END
```

Il risultato ottenuto è presentato in figura 3.4.

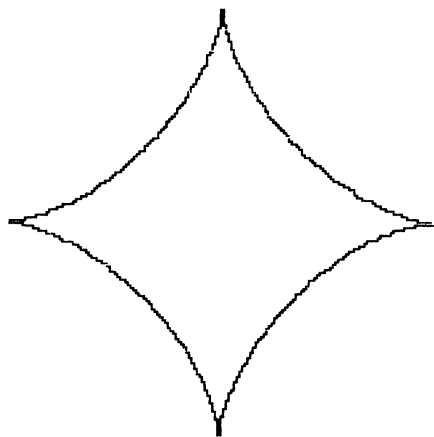



Fig. 3.4.

Il tracciamento di una circonferenza può essere effettuato secondo lo stesso principio. Il listing del programma è riprodotto qui di seguito e la figura 3.5 presenta il risultato ottenuto.

```
10 REM DISEGNO DI UN CERCHIO
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
```

```
50 XC=140:YC=80:SCALA=80
60 HGR:HCOLOR=3
70 HPLOT XC+SCALA,YC
80 FOR TETA=0.02 TO 6.28 STEP 0.02
90 X=XC+SCALA*COS(TETA)
100 Y=YC+SCALA*SIN(TETA)
110 HPLOT X,Y
120 VTAB 21: CALL -958: VTAB 22
130 PRINT TAB(14);"X= ";INT(X);TAB(23);"Y= ";INT(Y)
140 NEXT
150 END
```

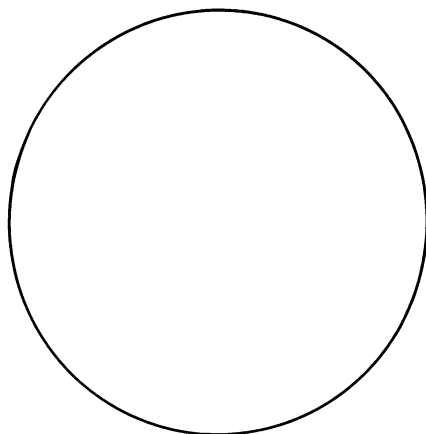



Fig. 3.5.

Le tre figure 3.3, 3.4 *e* 3.5 *sono state ottenute utilizzando il programma di copiatura dallo schermo residente sulla scheda d'interfaccia dell'*EPSON MXFT82.
Ecco un esempio di tracciamento di curva in coordinate cartesiane invece che in coordinate polari.

```
10  REM  PROGRAMMA DI TRACCIAMENTO CURVE
20  REM
30  REM  COPYRIGHT DE MERLY
40  REM
50  TEXT : HOME
60  PRINT "QUESTO PROGRAMMA TRACCIA DELLE CURVE"
70  PRINT "UTILIZZABILE CON LE FUNZIONI"
80  PRINT "MATEMATICHE DELL'APPLESOFT."
```

```
90  PRINT "QUINDI SCRIVETE LA VOSTRA ESPRESSIONE"
100  PRINT "CON IL SEGUENTE FORMATO:"
110  PRINT " 1010 Y=F(X)"
120  PRINT
130  PRINT " ESEMPIO: 1010 Y=EXP(X)"
140  PRINT : PRINT "SCRIVETE  RUN 170"
150  END
170  PRINT "SCRIVETE L'EQUAZIONE SECONDO LO STESSO FORMATO "
175  INPUT E$
180  DEF  FN I(X) =  INT (X + (1 -  SGN (X)) / 2)
184  REM
185  REM  PARAMETRI DELLA CURVA
186  REM
190  HOME
200  PRINT E$
210  PRINT : PRINT ""
220  INPUT "X INIZIALE -->";XD
230  PRINT
240  INPUT "X FINALE   -->";XF
250  ONERR  GOTO 2500
255  REM
256  REM  CALCOLO DELLA SCALA E DEGLI ASSI
257  REM
260  IF XF < XD THEN T = XF:XF = XD:XD = T
270  IF XD <  = 0 AND XF > 0 THEN XL = XF - XD:XC =  - XD
280  IF XD < 0 AND XF <  = 0 THEN XL =  - XD:XC =  - XD
290  IF XD > 0 AND XF > 0 THEN XL = XF:XC = 0
300 XI = XL / 279
310 XZ = XC / XI
320  PRINT : PRINT "OPERAZIONE IN CORSO"
330 YD = 0:YF = 0
340  FOR X = XD TO XF STEP XI
350  GOSUB 1010
360  IF Y > YF THEN YF = Y
370  IF Y < YD THEN YD = Y
380  NEXT X
390  IF YD <  = 0 AND YF > 0 THEN YL = YF - YD:YC =  - YD
400  IF YD < 0 AND YF <  = 0 THEN YL =  - YD:YC =  - YD
410  IF YD > 0 AND YF > 0 THEN YL = YF:YC = 0
420 YI = YL / 159
430 YZ = YC / YI
440 YZ = 159 -  INT (YZ)
441  REM
442  REM  VISUALIZZAZIONE DELL'EQUAZIONE
443  REM
450  HGR
460  VTAB 22
470  IF  LEN (E$) < 40 THEN  HTAB (40 -  LEN (E$)) / 2
480  PRINT E$
481  REM
```

```
482  REM  TRACCIAMENTO DEGLI ASSI
483  REM
490  HCOLOR= 3
500  HPLOT 0,YZ TO 279,YZ
510  HPLOT XZ,0 TO XZ,159
511  REM
512  REM  PUNTO DI PARTENZA
513  REM
520 X = XD
530  GOSUB 1010
540 Y = 159 - (Y + YC) / YI
550  HPLOT (X + XC) / XI,Y
551  REM
552  REM  TTRACCIAMENTO DELLA CURVA
553  REM
560  FOR X = XD + XI TO XF STEP XI
570  GOSUB 1010
580 Y = 159 - (YC + Y) / YI
590  HPLOT  TO (X + XC) / XI,Y
600  NEXT X
601  REM
602  REM  TRACCIAMENTO SCALA DELL'ASSE X
603  REM
610 R = XL
620  GOSUB 770
630  HCOLOR= 0
640  IF XD < 0 AND XF < 0 THEN XF = 0
650  IF XD > 0 AND XF > 0 THEN XD = 0
660  FOR X =  FN I(XD / DI) * DI TO  FN I(XF / DI) * DI STEP DI
670  HPLOT XZ + X / XI,YZ
680  NEXT X
690  VTAB 24: PRINT DI;"/ DIVISIONE PER X";
691  REM
692  REM  TRACCIAMENTO DELL'ASSE DELLE Y
693  REM
700 R = YL
710  GOSUB 770
720  FOR Y =  FN I(YD / DI) * DI TO  FN I(YF / DI) * DI STEP DI
730  HPLOT XZ,YZ - Y / YI
740  NEXT Y
750  PRINT  SPC( 2);DI;"/DIVISIONE PER Y";
760  GOTO 2650
761  REM
762  REM  DETERMINAZIONE DELLA SCALA
763  REM
770 K = 0
780  IF R < 1 THEN 840
790  IF R < 10 THEN 870
800 K = K + 1
810 R = R / 10
```

```
820  GOTO 790
830  IF R > 1 THEN 870
840 K = K - 1
850 R = R * 10
860  GOTO 830
870 DI =  INT (R) * 10 ^ (K - 1)
880  RETURN
1009  REM ***********************
1010 Y = 30
1011  REM ***********************
1020  RETURN
2400  REM
2410  REM  GESTIONE DEGLI ERRORI
2420  REM
2500  POKE 216,0: REM  ANNULLA ONERR
2510 EA =  PEEK (218) +  PEEK (219) * 256
2520 E2$ = "ERRORE"
2530 N$ = "":E1$ = ""
2540  IF EA <  > 1010 THEN N$ = "NON E'  ":E1$ = " ALLA LINEA " +
                                                          STR$ (EA)
2550  HOME : VTAB 21
2560 ER =  PEEK (222)
2570  IF E$ = "" THEN  PRINT "ERRORE NIENTE EQUAZIONE.": END
2580  PRINT "ERRORE DI SINTASSI ";N$;"NELL'EQUAZIONE."
2590  PRINT E$
2600  IF EA <  > 1010 THEN 2620
2610  IF ER = 16 OR ER = 163 OR ER = 224 THEN E2$ = "EQUAZIONE
                                                      SBAGLIATA"
2620  IF ER = 53 OR ER = 69 OR ER = 133 THEN E2$ = "DATI SBAGLIATI"
2630  IF ER = 255 THEN E2$ = "CONTROL-C"
2640  PRINT E2$ + E1$
2650  VTAB (15): PRINT  CHR$ (7): GET N$: TEXT
2660  END
```

3.4 FIGURE GRAFICHE AD ALTA RISOLUZIONE

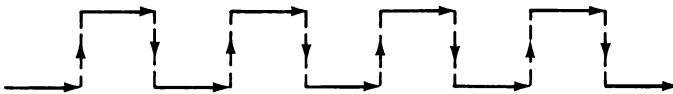
Utilizzando le funzioni grafiche a bassa od a alta risoluzione, dovete tracciare nuovamente ogni figura quando desiderate farla ruotare oppure quando volete modificarne la scala. Le figure grafiche ad alta risoluzione offrono tali possibilità.

3.4.1 Definizione delle figure grafiche

Una figura grafica ad alta risoluzione consiste di un insieme di otto vettori di spostamento:

- Spostamento del cursore di una posizione verso l'alto senza tracciamento.
- Spostamento del cursore di una posizione verso il basso senza tracciamento.
- Spostamento del cursore di una posizione verso destra senza tracciamento.
- Spostamento del cursore di una posizione verso sinistra senza tracciamento.
- Spostamento del cursore di una posizione verso l'alto con tracciamento del vettore sullo schermo.
- Spostamento del cursore di una posizione verso il basso con tracciamento del vettore sullo schermo.
- Spostamento del cursore di una posizione verso sinistra con tracciamento del vettore sullo schermo.
- Spostamento del cursore di una posizione verso destra con tracciamento del vettore sullo schermo.

Nel seguito di questo paragrafo disegneremo i "vettori senza tracciamento" con frecce e linee a tratto discontinuo e i "vettori con tracciamento" con frecce e linee a tratto continuo.

Esempio:

3.4.2 Codifica delle figure grafiche

Stabilito il disegno, dovete codificarlo. Per questo scopo, essendo i vettori numerati da zero a sette, è necessaria una codifica di tre bits per vettore.
Il primo bit rappresenta la scelta del tracciamento del vettore; gli altri due bits precisano lo spostamento come segue:

Vettore	Codice
→	01
↑	00
←	11
↓	10

La codifica degli otto vettori è riportata nella tabella a pagina seguente.
Dopo aver decomposto la figura in vettori e aver codificato questi ultimi, dovete raggruppare i vari codici nei bytes. Essendo un byte composto da otto bits, esso può contenere due codici di vettore (con/senza tracciamento)

Simbolo	Azione	Codice binario		Codice decimale
↑ (tratteggiato)	Spostamento verso l'alto senza tracciato	0	00	0
→ (tratteggiato)	Spostamento verso destra senza tracciato	0	01	1
↓ (tratteggiato)	Spostamento verso il basso senza tracciato	0	10	2
← (tratteggiato)	Spostamento verso sinistra senza tracciato	0	11	3
↑	Spostamento verso l'alto con tracciato	1	00	4
→	Spostamento verso destra con tracciato	1	01	5
↓	Spostamento verso il basso con tracciato	1	10	6
←	Spostamento verso sinistra con tracciato	1	11	7

e un codice di vettore senza tracciamento (sinistra, destra, basso) secondo il seguente formato:

Bit	7	6	5	4	3	2	1	0
	D	D	T	D	D	T	D	D

con D = bit di spostamento
T = bit di tracciamento.

Esaminiamo più da vicino la gestione dei bits 6 e 7. L'Applesoft ignora questi due bits se essi sono uguali a zero e considera questo vettore come di tracciamento se almeno uno dei due bits è uguale a uno. Potete quindi utilizzare questi due bits per codificare i vettori da uno a tre.
Il termine di una figura grafica è indicata da un byte nullo.

3.4.3 Raggruppamento in una tabella delle figure grafiche

Per permettere il funzionamento delle istruzioni dell'Applesoft, le figure grafiche ad alta risoluzione devono essere raggruppate in una tabella che potrà essere memorizzata su cassetta o dischetto. L'organizzazione di una tabella segue il formato seguente:

Numero di byte	Significato	
0	Numero n di figure nella tabella (da 0 a 255)	
1	inutilizzate	
2	Bits di peso minore	spostamento della figura n 1 nella tabella (numero del primo byte della figura 1)
3	Bits di peso maggiore	
4	Spostamento della figura 2	
5		
.		
.		
.		
2i + 0	Spostamento della figura i	
2i + 1		
.		
.		
.		
2n + 1	Spostamento della figura n	
2	Definizione della figura grafica n 1	
	00 (termine della definizione)	
	.	
	.	
	.	
	Definizione della figura n	
	00 (termine della figura n)	

Nel caso che la tabella comprenda una sola figura, essa avrà il seguente formato:

Byte	Valore
0	1
1	0
2	4
3	0
4	Definizione
.	della figura
.	00

*Per evitare che l'*Applesoft *distrugga la tabella creata, vi consigliamo di porla in memoria al di sopra di* HIMEM.

3.4.4 Programma di creazione di una tabella di figure grafiche

Ecco un esempio che permette di acquisire i vettori di spostamento, di raggrupparli in una tabella, di memorizzare quest'ultima su dischetto ed infine di visualizzare sullo schermo le figure ottenute:

Righe da 10 a 999: programma principale
- da 100 a 250: inizializzazioni
- da 260 a 360: acquisizione del numero di figure della tabella e controllo
- da 370 a 530: acquisizione dell'indirizzo di inizio della tabella, controllo e conversione in base decimale
- da 535 a 600: creazione del catalogo della tabella
- da 605 a 640: definizione delle figure
- da 650 a 750: salvataggio su dischetto se necessario
- da 800 a 999: prova delle figure grafiche
- da 1000 a 1440: sottoprogramma di acquisizione dei vettori di una figura
- da 1997 a 2040: sottoprogramma di ricopiatura dei tre vettori in memoria (codifica di un byte)
- da 3000 a 3080: sottoprogramma di validità di una stringa numerica.
- da 3100 a 3230: sottoprogramma di controllo della correttezza di una stringa esadecimale e conversione in base decimale
- da 10000 a 10001: dati di conversione da base esadecimale a base decimale.

```
10 REM CREAZIONE E PROVA DI UNA TABELLA DI FIGURE
20 REM
30 REM COPYRIGHT DE MERLY
40 REM
100 HOME:VTAB 2
110 PRINT "---------------------------------------"
120 PRINT
130 PRINT "     CREAZIONE DI UNA TABELLA DI FIGURE"
140 PRINT
150 PRINT "---------------------------------------"
160 REM
170 REM INIZIALIZZAZIONI
180 REM
200 DIM TCHD$(16,2),CDVETT(3)
210 FOR I=1 TO 16
220 FOR J=1 TO 2
```

```
230 READ TCHD$(I,J)
240 NEXT:NEXT
250 POKE 34,7
260 REM
270 REM LUNGHEZZA DELLA TABELLA
280 REM
290 FOR I=1 TO 1000:NEXT:VTAB 9:CALL -958:VTAB 11:T1=0
300 INPUT"NUMERO DELLE FIGURE NELLA TABELLA";NB$
310 C$=NB$:Y=9:GOSUB 3000
350 ON T1 GOTO 290
360 NB=VAL(NB$)
370 REM
380 REM INDIRIZZO D'INIZIO
390 REM
400 VTAB 13:PRINT "---------------------------------------"
410 FOR I=1 TO 1000:NEXT:VTAB 15:CALL -958:VTAB 17:T1=0
420 INPUT "INDIRIZZO D'INIZIO";ADR$
430 REM
440 REM VALORE DECIMALE O ESADECIMALE?
450 REM
460 T2=1
470 IF LEFT$(ADR$,1)="$" THEN T2=2
480 T1=0
485 Y=15:C$=ADR$
490 ON T2 GOSUB 3000,3100
500 ON T1 GOTO 410
510 DEF FN MOD(X)=INT((X/256-INT(X/256))*256+0.05)*SGN(X/256)
520 ADR=S
530 VTAB 17:HTAB 19:PRINT ADR
540 POKE 232,FN MOD(ADR)
550 POKE 233,INT(ADR/256)
560 POKE ADR,NB
570 POKE ADR+1,0
580 POKE ADR+2,FN MOD(2*NB+2)
590 POKE ADR+3,INT((2*NB+2)/256)
600 ACR=ADR+2*NB+2
610 FOR K=1 TO NB
620 GOSUB 1000
630 IF K< >NB THEN POKE ADR+2*K+2,FN MOD(ACR-ADR):POKE ADR+2*K+3
                                                   ,INT((ACR-ADR)/256)
640 NEXT K
650 REM
660 REM SALVATAGGIO DELLA TABELLA?
670 REM
680 CALL -936:VTAB 10
690 PRINT"DESIDERATE IL SALVATAGGIO DELLA TABELLA CREATA?";
700 GET T$
710 IF T$<>"O" AND T$<>"N" THEN 700
720 IF T$="N" THEN 800
730 INPUT "NOME DEL FILE DI SALVATAGGIO:";T$
```

```
740 T$="BSAVE "+T$+",A"+STR$(ADR)+",L"+STR$(ACR-ADR)
750 PRINT CHR$(4);T$
800 REM
810 REM PROVA DELLE FIGURE
820 REM
830 VTAB 12:PRINT "---------------------------------------":
                                                         VTAB 14
840 PRINT "DESIDERATE PROVARE LE FIGURE?"
850 GET T$
860 IF T$<>"O" AND T$<>"N" THEN 850
870 IF T$="N" THEN 950
880 HGR:HCOLOR=3:SCALA=1:ROT=0
890 FOR K=1 TO NB
900 DRAW K AT 140,80
910 FOR I=1 TO 5000:NEXT
920 XDRAW K AT 140,80
930 NEXT
950 POKE 34,0:TEXT:HOME
999 END
1000 REM
1010 REM PRELEVAMENTO DEI VETTORI DI UNA FIGURA
1020 REM
1030 REM E CODIFICA NEI BYTES
1040 REMCHE SI TROVANO ALL'INDIRIZZO CORRENTE ADR
1050 REM
1060 FOR I=1 TO 1000:NEXT:HOME
1070 VTAB 9:PRINT"FIGURA NUMERO";K
1080 VTAB 11:PRINT "---------------------------------------"
1090 V=0:NO=0:POKE 34,11:VTAB 13
1095 V=V+1
1100 HTAB 8:PRINT "VETTORE NUMERO ";V;" (0-7)?";
1120 GET T$:IF ASC(T$)<48 OR ASC(T$)>55 THEN 1120
1130 T=VAL(T$):PRINT T
1140 ON T GOTO 1250,1250,1250,1400,1400,1400,1400
1150 REM
1160 REM CASO DI UN VETTORE NULLO
1170 REM
1180 NO=NO+1:CDVETT(NO)=0:NZ=NZ+1
1190 IF NO=3 THEN GOSUB 2000:CDVETT(NO)=0
1200 IF NZ=3 THEN PRINT:INVERSE:POKE ACR,0:ACR=ACR+1:PRINT "FINE
     DELLA FIGURA"; K:NORMAL:POKE 34,7:FOR I=1 TO 1000:NEXT:RETURN
1210 GOTO 1095
1220 REM
1230 REM CASO DI UN VETTORE SENZA TRACCIA
1240 REM
1250 NO=NO+1:CDVETT(NO)=T
1260 IF NO=3 THEN GOSUB 2000
1270 NZ=0
1280 GOTO 1095
1370 REM
```

```
1380 REM CASO DI UN VETTORE AVENTE TRACCIA
1390 REM
1400 NO=NO+1
1410 IF NO=3 THEN CDVETT(NO)=0:GOSUB 2000
1420 CDVETT(NO)=T
1430 NZ=0
1440 GOTO 1095
1997 REM
1998 REM COPIA DI TRE VETTORI IN MEMORIA
1999 REM
2000 NN=CDVETT(1)+CDVETT(2)*8+CDVETT(3)*64
2010 POKE ACR,NN
2020 ACR=ACR+1
2030 NO=1
2040 RETURN
3000 REM
3010 REM TEST DI VALIDITA' DI UNA STRINGA NUMERICA
3020 REM
3030 FOR I=1 TO LEN(C$)
3040 T=ASC(MID$(C$,I,1))
3050 IF T<48 OR T>57 THEN VTAB Y:PRINT CHR$(7);TAB(8);:INVERSE:
     PRINT CHR$(T);   "E' UN CARATTERE ILLEGALE":NORMAL:T1=1:I=LEN(C$)
3060 NEXT
3070 S=VAL(C$)
3080 RETURN
3100 REM
3110 CONVERSIONE DA ESADECIMALE A DECIMALE
3120 REM
3130 S=0:C$=MID$(C$,2)
3140 FOR I=1 TO LEN(C$)
3150 T=ASC(MID$(C$,I,1))
3160 IF T<48 OT T>70 THEN VTAB Y:HTAB 8:INVERSE:PRINT CHR$(T);
     "E' UN CARATTERE ILLEGALE":NORMAL:T1=1:I=LEN(C$):GOTO 3220
3170 T$=CHR$(T)
3180 FOR J=1 TO 16
3190 IF T$=TCHD$(J,1) THEN T=VAL(TCHD$(J,2)):J=16
3200 NEXT
3210 S=S+16^(LEN(C$)-I)*T
3220 NEXT
3230 RETURN
5000 REM
5010 REM GESTIONE DEGLI ERRORI
5020 REM
5030 STOP
5999 RETURN
10000 DATA 0,0,1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8,9,9
10001 DATA "A",10,"B",11,"C",12,"D",13,"E",14,"F",15
```

3.4.5 Salvataggio su cassetta

Per ricopiare su cassetta la vostra tabella, calcolate la sua lunghezza x e scrivete sulla tastiera le seguenti istruzioni:

] CALL-151 passaggio al Monitor
* 00:0x 00 posizionamento della lunghezza all'indirizzo zero
- fate partire la cassetta (PLAY)
- scrivete 0.1W.

La tabella è a questo punto memorizzata sulla cassetta. Due "bip" segnalano la fine corretta della memorizzazione.
Se durante la memorizzazione compare un messaggio d'errore, riprendete le operazioni dall'inizio e, se non ottenete un risultato positivo, consultate il capitolo 4 *dedicato al Monitor* (par. 4.3: I comandi del Monitor).
Per rileggere la cassetta e caricare la tabella delle figure in memoria, il Basic Applesoft mette a disposizione il comando SHLOAD. All'atto dell'utilizzo di SHLOAD, HIMEM viene diminuito della lunghezza della tabella e quest'ultima viene posta nella zona di memoria così liberata.

3.4.6 Salvataggio su dischetto

Il comando di salvataggio di una tabella su dischetto è il medesimo che viene utilizzato per un sottoprogramma. Utilizzerete l'istruzione

BSAVE nome del file, Aa, L1

dove a è l'indirizzo decimale della tabella in memoria
1 è la lunghezza della tabella

per salvare la tabella su dischetto e

BLOAD nome del file, Aa

dove a è l'indirizzo della tabella

poi utilizzerete POKE 232, byte di peso minore dell'indirizzo
POKE 233, byte di peso maggiore dell'indirizzo

Queste due istruzioni del tipo POKE *permettono di precisare all'*Applesoft *l'indirizzo della tabella delle figure.*

3.4.7 Istruzioni di lavoro sulle figure grafiche

L'Applesoft mette a disposizione le quattro istruzioni seguenti che permettono di disegnare, di cancellare, di orientare, di ingrandire o di rimpicciolire una figura grafica sullo schermo:

DRAW disegno di una figura
XDRAW cancellazione di una figura

ROT orientazione (rotazione) della figura sullo schermo
SCALE modifica della scala della figura sullo schermo.

Queste istruzioni utilizzano la pagina grafica ad alta risoluzione scelta (1 o 2) e il colore selezionato dall'ultima istruzione HCOLOR.

• L'istruzione DRAW

Essa permette di tracciare sullo schermo una figura della tabella individuata dal suo numero, nel colore la scala e l'orientamento scelti precedentemente. Il formato dell'istruzione è il seguente:

```
DRAW        4        AT        125,90
            ^                  ^   ^
            ↑                  ↑   ↑
        (numero            (colonna, riga
         nella              d'inizio)
         tabella)
```

È possibile non precisare le coordinate di inizio della figura ; essa viene allora tracciata a partire dall'ultimo punto raggiunto sullo schermo.
Se non è stato raggiunto nessun punto, viene scelta l'origine degli assi.
Se la tabella non è stata caricata correttamente o se vi siete dimenticati di precisare l'indirizzo all'Applesoft, vedrete visualizzato il seguente messaggio d'errore:

ILLEGAL QUANTITY ERROR

• L'istruzione XDRAW

Quest'istruzione vi permette di cancellare una figura grafica dallo schermo. La sua sintassi è identica a quella dell'istruzione DRAW.

XDRAW 12 AT 30,15

XDRAW visualizza in effetti la figura nel colore complementare al colore effettivo (negativo).

Colore	**Colore complementare**
Nero	Bianco
Bianco	Nero
Viola	Verde
Verde	Viola
Arancione	Blu
Blu	Arancione

• L'istruzione SCALE=

Essa permette di precisare la scala con la quale la figura grafica sarà disegnata. SCALE deve sempre essere stata eseguita almeno una volta prima di DRAW. Può essere usata direttamente o da programma.

SCALE = 25

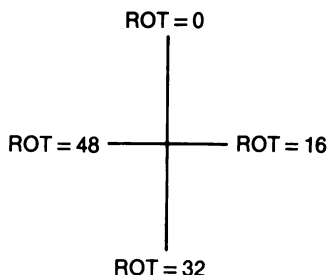
Il valore della scala è compreso tra 1 (un punto per vettore) e 255 (255 punti per vettore). Il valore zero corrisponde a 256 punti per vettore.
La parola SCALE *non è una parola chiave riservata per l'*Applesoft. *Viene riconosciuta solo* SCALE = .

- **L'istruzione ROT**

Essa permette di fare ruotare gli assi della figura rispetto all'asse verticale. Ha il seguente formato:

ROT = valore

dove il valore è un numero compreso tra 0 e 255 pur essendo trattati 64 valori (da 0 a 63). L'Applesoft utilizza sempre il massimo valore possibile inferiore al valore indicato.
Il numero di valori possibili dipende in effetti dalla scala nei termini seguenti:

Scala	Numero di valori
1	4 (0,16,32,48)
2	8
3	16
4	32
> = 5	64 (da 0 a 63)

Come per l'istruzione SCALE = , ROT *non è una parola riservata.*

3.4.8 Utilizzo delle figure grafiche

Ecco un programma che crea un effetto di zoom su qualsiasi figura grafica la cui grandezza sia inferiore a quella di un quadrato di 10 punti per lato.

```
10 REM EFFETTO ZOOM
20 REM
30 REM COPYRIGHT DE MERLY-LAMOITIER
40 REM
50 HOME
60 INPUT "NOME DEL FILE CONTENENTE LA TABELLA";NOM$
70 PRINT
80 INPUT "INDIRIZZO DI CARICAMENTO (DECIMALE)";ADR
90 PRINT CHR$(4);"BLOAD "+NOM$+",A"+STR$(ADR)
130 DEF FN MOD(X)=INT((X/256-INT(X/256))*256+0.05)*SGN(X)
140 POKE 232,FN MOD(ADR)
150 POKE 233,INT(ADR/256)
160 HGR:POKE -16302,0:HCOLOR=3
170 FOR I=1 TO PEEK(ADR)
180 FOR J=1 TO 64
190 ROT=J
200 SCALA=1+J/2
210 DRAW I AT 140,120
220 NEXT
```

La figura 3.6 rappresenta tale effetto su un quadrato i cui lati sono di lunghezza uno (1 punto), cambiando la riga 200 con:

200 SCALE = 10 + J/2

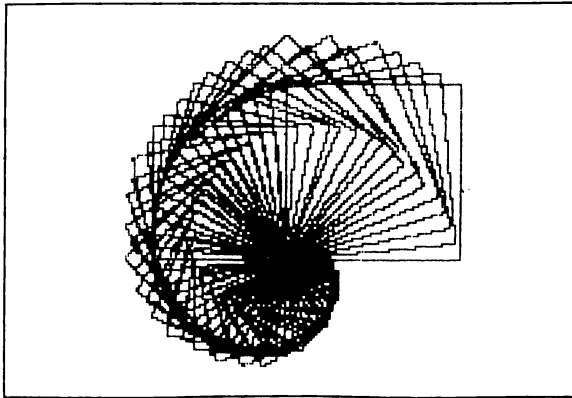



Fig. 3.6.

3.5 PERIFERICHE GRAFICHE

3.5.1 Periferiche di uscita

Esistono due tipi principali di periferiche grafiche per l'Apple II. Esse sono le stampanti grafiche e le tavolette grafiche.

La definizione dell'immagine ottenuta è di:

- 480 punti per riga, per la stampante termica SILENTYPE (Apple);
- 480 punti per riga con funzionamento normale o 960 punti per riga con funzionamento a doppia densità, per la stampante EPSON (precisione di 5/10 di mm o 2,5/10 di mm).

Per una tavoletta grafica a bassa gamma collegabile all'Apple tramite un'interfaccia parallela, la precisione può raggiungere 1/10 di mm.
Schede di questo tipo non modificano la definizione dello schermo.

3.5.2 Periferiche di ingresso

- Scheda DIGISECTOR, che permette di ricevere le immagini provenienti da una cinepresa, di digitalizzarle e di trattarle, per esempio, in applicazioni di riconoscimento di forme.
- Tavoletta grafica (Apple) per tracciare un disegno con l'aiuto di uno stiletto magnetico, digitalizzarlo e disporlo in una pagina di memoria ad alta risoluzione al fine di riprodurlo sullo schermo.
- Sistema a tre dimensioni il cui scopo è di determinare le coordinate spaziali dell'estremità di uno stiletto, di seguire i contorni degli oggetti e di rappresentarli in prospettiva 3D sullo schermo dell'Apple II (Micro control systems).

Torneremo ad occuparci di queste schede nel volume 2.

3.6 POSSIBILITÀ SONORE DELL'APPLE II

Sotto la tastiera dell'Apple II si trova un altoparlante gestito dal Monitor. Il suo impiego è molto semplice: dovete definire le note da emettere, facendone variare la frequenza e la durata.

3.6.1 Messa in opera

L'altoparlante utilizza l'indirizzo $C030 (—16336). Qualsiasi accesso a questo indirizzo provoca l'emissione di un "bip".
In Basic, la funzione da utilizzare è PEEK (—16336). La frequenza ottenuta è debole. Vi presentiamo qui di seguito un esempio di sottoprogramma in assembler di gestione dell'altoparlante.

3.6.2 Emissione di una nota di durata e frequenza date

Diversi sottoprogrammi del Monitor gestiscono l'altoparlante per determinate frequenze. Essi sono descritti al paragrafo 4.3.4.
L'esempio proposto richiama il sottoprogramma d'attesa del Monitor WAIT posto all'indirizzo $FCA8 (—856).

```
0300            1         ORG $300
0300            2  WAIT   EQU $FCA8         ; ATTESA RITARDO
0300            3  HP     EQU $C030         ; ALTOPARLANTE
0301            4  FREQ   DFS $1
0302            5  DUREE  DFS $1
0302 AC0003     6  NOTE   LDY FREQ
0305 AE0003     7  ET1    LDX FREQ
0308 A904       8         LDA #$04
030A 20A8FC     9         JSR WAIT
030D AD30C0    10         LDA HP
0310 E8        11  ET2    INX
0311 D0FD      12         BNE ET2
0313 88        13         DEY
0314 D0EF      14         BNE ET1
0316 CE0103    15         DEC DUREE
0319 D0E7      16         BNE NOTE
031B 60        17         RTS
               18         END
```

Listato dell'assembler: generazione di un suono

Per utilizzare questo sottoprogramma in Basic, è sufficiente:

- porre agli indirizzi
  - $300 frequenza (0—255)
  - $301 durata (0—255)
- eseguire l'istruzione
  - CALL 700 ($302)

Esempio:

```
5  PRINT "BLOAD BSON"
10  FOR I = 1 TO 255
20  POKE 768,I
25  POKE 769,1
30  CALL 770
40  NEXT
```

3.7 MANOPOLE PER GIOCHI E JOYSTICK

Le manopole per giochi vi permettono di fornire all'Apple II dei dati continui (analogici) senza bisogno di scrivere sulla tastiera.
Ogni manopola è composta da un potenziometro regolabile (bottone centrale) e da un interruttore (bottone a molla). Quest'ultimo permette per esempio di convalidare i dati letti sulle manopole.
In Basic, la funzione PDL(I) legge la manopola numero I (0—1); come vi verrà spiegato nel capitolo 4, tra ogni lettura di un valore su una qualsiasi

delle manopole deve trascorrere un tempo minimo che corrisponde approssimativamente a quello stabilito dalle istruzioni:

FOR I=1 TO 10:NEXT I

Esaminate l'esempio del paragrafo 3.2.4. Queste istruzioni non servono apparentemente a nulla, ma in effetti sono indispensabili.
L'accesso al bottone (a molla) è conseguito tramite la funzione PEEK agli indirizzi $C061 (manopola 0) e $C062 (manopola 1). Se il valore ottenuto è maggiore di 128, viene premuto il bottone corrispondente.
La Joystick è l'equivalente delle manopole per giochi per l'Apple II.
Per l'utilizzatore, essa è composta di un "manico" che può spostarsi in tutte le direzioni (360 gradi). È molto pratica nei giochi in cui il movimento è piano (modifica simultanea delle coordinate X e Y).

CAPITOLO 4

La programmazione in assemblatore 6502 e il monitor dell'Apple II

4.1 INTRODUZIONE

Nelle memorie ROM dell'Apple risiedono permanentemente il linguaggio Basic (Integer Basic o Applesoft) e un programma di controllo chiamato Monitor. Questo programma è scritto in linguaggio assembler e consente le operazioni standard d'ingresso-uscita (gestione dello schermo e della tastiera). Esso permette inoltre di generare dei programmi in linguaggio assembler, di interrogare e di modificare il contenuto della memoria ecc.
Una volta descritti il microprocessore 6502 (struttura interna e istruzioni) e le tecniche di base della programmazione in assembler, studieremo via via il monitor, i suoi comandi, i sottoprogrammi accessibili all'utente e la scheda di memoria. Data l'efficienza del monitor dell'Apple II, risulta molto utile una sua profonda conoscenza onde poter lavorare efficacemente in assembler e in Basic.

4.2 LA PROGRAMMAZIONE IN ASSEMBLATORE 6502

4.2.1 Introduzione

Il possessore di un Apple II che desidera realizzare dei programmi di buona qualità, si interesserà necessariamente del linguaggio assembler del microprocessore 6502. In effetti, il Basic deve essere utilizzato per applicazioni limitate e non ripetitive, però risulta esasperante a causa della sua lentezza quando esegue dei calcoli molto ripetitivi. Potreste allora pensare di eseguire i vostri programmi in Pascal, Lisp o Forth. Ma per dei piccoli sottoprogrammi risparmierete tempo e denaro programmando in assembler 6502. Vi sarà a questo punto necessario un assemblatore (programma che traduce il linguaggio mnemonico in linguaggio macchina). Vi sono offerte molte possibilità:

- Utilizzare il miniassemblatore in ROM del vecchio Monitor. L'interesse sta nel fatto che il suo avviamento è molto facile, ma non potrete conservare i commenti.
- Recuperare il miniassemblatore nel file INTBASIC fornito con il DOS 3.3.
- Comprare il miniassemblatore su dischetto per l'Apple II.
- Acquistare un assemblatore potente come il LISA, che presenta l'enorme vantaggio di scovare gli errori di sintassi quando inserite le istruzioni (come nell'Integer Basic). Esso può quindi lavorare molto velocemente.

L'obiettivo non è quello di insegnarvi a maneggiare questo o quell'assemblatore.
Il lettore interessato potrà riferirsi al paragrafo 4.3.3 relativamente al miniassemblatore o alla documentazione corrispondente al LISA.
Ora descriveremo la struttura del microprocessore 6502, le sue specifiche in confronto agli altri microprocessori a otto bits esistenti e infine i suoi codici mnemonici.

4.2.2 Il microprocessore 6502

Esso ha nell'Apple II la funzione di una unità centrale su un solo chip. Esso comprende un'unità aritmetico-logica (UAL) per l'esecuzione dei calcoli e un'unità di controllo incaricata di sincronizzare il sistema Apple II; esso comunica con gli altri chips mediante un insieme di fili denominati bus. Questi ultimi sono in numero di tre:

- il bus di indirizzi a 16 fili che consente di selezionare una cella di memoria fra le 65.536 che compongono la memoria complessiva;
- il bus dati a 8 fili (8 bits) che consente il trasferimento dei dati tra i diversi chips;
- il bus comandi utilizzato per la sincronizzazione dell'intero sistema.

Aprendo il vostro Apple II, vedrete quanto segue:

- il microprocessore 6502;
- 8 chips di memoria RAM (lettura e scrittura di informazioni);
- due memorie ROM (a sola lettura) che contengono il Monitor e il Basic;
- 7 connettori per le schede di espansione;
- altri chips gestenti lo schermo e la tastiera... (cfr. foto C. cap. 1).

Il microprocessore 6502 ha la seguente struttura:

- una unità aritmetico-logica (UAL);
- un registro chiamato accumulatore (A) di 8 bits;
- due registri indice X e Y di 8 bits;
- un puntatore di pila S di 9 bits;
- un registro di indicatore di stato P;
- un program-counter (PC) di 16 bits che punta alla successiva istruzione da eseguire.

• **L'accumulatore**

L'UAL è collegata a un registro speciale, l'accumulatore. Essa fa riferimento automaticamente a questo registro come ad uno dei suoi ingressi. Tuttavia esiste la possibilità di bypassarlo. Nelle operazioni aritmetiche o logiche uno degli operandi sarà in una cella di memoria mentre l'altro sarà nell'accumulatore. Il risultato sarà in seguito immagazzinato nell'accumulatore.

6502	**INTEL 8080-ZILOG Z80**
Istruzioni molto rapide ma poco potenti	Istruzioni più lunghe in dimensione e durata, ma potenti

Questa architettura interna è classica: consente di avere delle istruzioni brevi di lunghezza 1 byte, di esecuzione molto rapida. D'altra parte l'inconveniente che si presenta consiste nel fatto che i dati devono essere preliminarmente caricati. Esaminiamo ora in maggiore dettaglio il ruolo che viene assunto dai vari registri singolarmente.

• **Il program-counter**

Supponiamo che dopo l'esecuzione di un'istruzione il PC contenga l'indirizzo su 16 bits della prossima istruzione da eseguire. Eseguiamo questa e osserviamo quello che accade passo per passo.
Ogni processore esegue tre fasi:

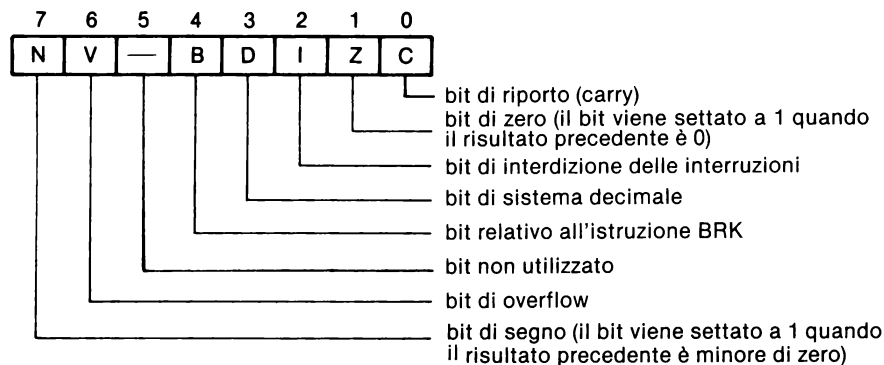
Ricerca della prossima istruzione da eseguire. Il contenuto del PC viene collocato sul bus indirizzi. La memoria selezionata deposita l'istruzione sul bus dati.

Decodifica ed esecuzione. Il microprocessore esamina l'istruzione per generare il segnale di controllo. Si dice che esso la "decodifica". A seconda dell'istruzione, tutto si può svolgere nel 6502 oppure vengono anche fatti degli accessi in memoria, il che spiega le differenze di tempo necessarie per il trattamento.

Ricerca della prossima istruzione. In seguito ad ogni accesso alla memoria, per ricercare i bytes corrispondenti all'istruzione (uno per il codice dell'istruzione, uno o due per gli operandi), il PC viene incrementato. In questo modo, alla fine di ogni istruzione, esso contiene sempre l'indirizzo della successiva.

• **Il registro dell'indicatore di stato P**

Esso comprende 8 bits di indicatori di stati (flags). Ogni bit viene utilizzato per rappresentare una condizione particolare (uguaglianza o no, zero o no, positivo o no, riporto o no...).

Questi bits vengono testati all'atto dell'esecuzione di istruzioni di salto condizionato (BMI, BPL, BVC, BCC, BCS, BEQ, BVS,...).
L'istruzione IF condiziona GOTO del Basic si ottiene posizionando il bit corrispondente alla condizione e testando questo bit tramite l'istruzione di salto condizionato corrispondente.

Contenuto del registro di stato

• La pila e il puntatore di pila S. L'impaginazione

La pila è un insieme di registri o di celle di memoria organizzata mediante una struttura LIFO (Last In First Out). L'ultimo elemento introdotto è sempre in cima alla pila (il paragone è semplice se pensiamo ad esempio a una pila di piatti). La presenza di una pila è praticamente indispensabile nella gestione di sottoprogrammi, degli interrupts, o per immagazzinare una grande quantità di dati temporaneamente.
L'approccio seguito dal 6502 è quello di una pila programmabile (celle di memoria).
Il registro S contiene l'indirizzo della cima della pila. A differenza dei microprocessori INTEL 8080 o ZILOG Z80, la pila del 6502 è di dimensione finita. Il registro S, disponendo di 9 bits, può indirizzare una memoria compresa tra 256 e 511, avendo il bit più significativo settato a 1.
La pila corrispondente alla pagina 1 della memoria del 6502. Questa è suddivisa in pagine di 256 bytes cadauna. Questo è il concetto dell'impaginazione. Vedremo nel paragrafo 4.3.4 "descrizione della mappa di memoria" l'utilizzo delle diverse pagine nell'Apple II + .

• Metodi di indirizzamento nel 6502. Registri indice X e Y

Il microprocessore 6502 ammette i seguenti metodi di indirizzamento:

- indirizzamento implicito: selezione dei diversi registri utilizzati dall'istruzione;

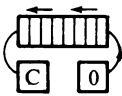
- indirizzamento immediato: caricamento di un registro mediante un valore numerico definito precedentemente;
- indirizzamento assoluto: indirizzi su 16 bits;
- indirizzamento in pagina zero: indirizzi su 8 bits riservati ai dati ai quali si vuole accedere molto rapidamente;
- indirizzamento relativo: l'indirizzo indicato è relativo ad un inizio di pagina;

15 8	7 0
numero di pagina	indirizzo relativo

- indirizzamento a indice: il 6502 non dispone di un indirizzamento a indice del tutto generale. Esso possiede due registri indice X e Y di 8 bits. Il contenuto di un registro indice viene aggiunto, se necessario, alla parte indirizzo dell'istruzione. I due registri permettono di accedere molto facilmente ai diversi elementi di una tabella; esistono all'uopo delle istruzioni per l'incremento o il decremento di questi registri e per controllare i valori in essi contenuti in rapporto a quelli delle celle di memoria (criteri di arresto).

L'esempio seguente mostra come si esegue il trasferimento di N bytes tra gli indirizzi DEP e ARR (inizio di blocco):

```
        LDX # N      ; caricamento del numero nel registro X
        LDA DEP,X    ; lettura
PROCH   STA ARR,X    ; ricopiatura
        DEX          ; X=X—1
        BNE PROCH    ; salto se non terminato
```

- indirizzamento indiretto: esso utilizza sempre la pagina zero e deve essere a indice sia prima dell'indirettezza, sia dopo.

4.2.3 Tabella delle istruzioni del 6502

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
ADC	A←A+M+C	immediato	ACD # OPER	69 XX	N
addizione		pagina zero	ADC OPER	65 XX	Z
dell'accumulatore e		pagina zero, X	ADC OPER, X	75 XX	V
del contenuto di		assoluto	ADC OPER	6D XXXX	
una cella di		assoluto, X	ADC OPER, X	7D XXXX	
memoria e del		assoluto, Y	ADC OPER, Y	79 XXXX	
riporto (carry)		(indiretto, X)	ADC (OPER, X)	61 XX	
		(indiretto), Y	ADC (OPER), Y	71 XX	

(segue)

(seguito)

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
AND e logico dell'accumulatore e del contenuto di una cella di memoria	A←AM	immediato pagina zero pagina zero, X assoluto assoluto, X assoluto, Y (indiretto, X) (indiretto), Y	AND # OPER AND OPER AND OPER, X AND OPER AND OPER, X AND OPER, Y AND (OPER, X) AND (OPER), Y	29 XX 25 XX 35 XX 2D XXXX 3D XXXX 39 XXXX 21 XX 31 XX	N Z
ASL spostamento logico verso la sinistra dell'accumulatore o del contenuto di una cella di memoria	C ← [ ] ← 0	accumulatore pagina 0 pagina 0, X assoluto assoluto, X	ASL A ASL OPER ASL OPER, X ASL OPER ASL OPER, X	0A 06 XX 16 XX 0E XXXX 14 XXXX	C N Z
BCC salto se non c'è riporto	salto se C = 0	relativo	BCC OPER	90 XX	
BCS salto se c'è riporto	salto se C = 1	relativo relativo	BCS OPER	B0 XX	
BEQ salto se risultato nullo	salto se Z = 1	relativo	BEQ OPER	F0 XX	
BIT tests di bits della memoria con l'accumulatore	Z←$\overline{A \wedge M}$ N←M7 V←M6	pagina zero assoluto	BIT OPER BIT OPER	24 XX 2C XXXX	N = M7 V = M6 Z = $\overline{A \wedge M}$
BMI salto se risultato negativo	salto se Z = 1	relativo	BMI OPER	30 XX	
BNE salto se risultato non nullo	salto se Z = 0	relativo	BNE OPER	D0 XX	
BPL salto se risultato positivo	salto se N = 0	relativo	BPL OPER	10 XX	
BRK creazione di un'interruzione	interruzione	implicito	BRK	00	I←1
BVC salto se non si è prodotto alcun superamento	salto se V = 0	relativo	BVC OPER	50 XX	

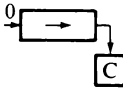
(segue)

(seguito)

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
BVS salto se si è prodotto un superamento	salto se V = 1	relativo	BVS OPER	70 XX	
CLC rimessa a zero del bit di riporto	C←0	implicito	CLC	18	C←0
CLD rimessa a 0 del bit decimale	D←0	implicito	CLD	D8	D←0
CLI autorizzazione delle interruzioni	I←0	implicito	CLI	58	I←0
CLV rimessa a 0 del bit di superamento (overflow)	V←0	implicito	CLV	B8	V←0

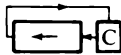
CMP
confronto tra i contenuti dell'accumulatore e di una cella di memoria (65021 fa A-M e posiziona i bits N, Z, C, secondo il risultato, esso non accede all'accumulatore in scrittura)
	A-M	immediato
		pagina zero
		pagina zero, X
		assoluto
		assoluto, X
		assoluto, Y
		(indiretto, X)
		(indiretto), Y
			CMP # OPER
			CMP OPER
			CMP OPER, X
			CMP OPER
			CMP OPER, X
			CMP OPER, Y
			CMP (OPER, X)
			CMP (OPER), Y
				C9 XX
				C5 XX
				D5 XX
				CD XXXX
				DD XXXX
				D9 XXXX
				C1 XX
				D1 XX
					N
					Z
					C

CPX confronto tra i contenuti dei X e una cella di memoria	X-M	immediato pagina zero assoluto	CPX # OPER CPX OPER CPX OPER	E0 XX E4 XX EC XXXX	N Z C
CPY confronto tra i contenuti di Y e una cella di memoria	Y-M	immediato pagina zero assoluto	CPY # OPER CPY OPER CPY OPER	C0 XX C4 XX CC XXXX	N Z C
DEC decremento del contenuto di una cella di memoria di un'unità	M←M—1	pagina zero pagina zero, X assoluto assoluto, X	DEC OPER DEC OPER, X DEC OPER DEC OPER, X	C6 XX D6 XX CE XXXX DE XXXX	N Z

(segue)

(seguito)

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
DEX decremento di un'unità del registro	X←X−1	implicito	DEX	CA	N Z
DEY decremento di un'unità del registro	Y←Y−1	implicito	DEY	88	N Z
EOR or esclusivo logico tra i contenuti dell'accumulatore e di una cella di memoria	A←A⊕M	immediato	EOR # OPER	49 XX	N Z
		pagina zero	EOR OPER	45 XX	
		pagina zero, X	EOR OPER, X	55 XX	
		assoluto	EOR OPER	4D XXXX	
		assoluto, X	EOR OPER, X	5D XXXX	
		assoluto, Y	EOR OPER, Y	59 XXXX	
		(indiretto, X)	EOR (OPER, X)	41 XX	
		(indiretto), Y	EOR (OPER), Y	51 XX	
INC incremento di un'unità del contenuto di una cella di memoria	M←M+1	pagina zero	INC OPER	E6 XX	N Z
		pagina zero, X	INC OPER, X	F6 XX	
		assoluto	INC OPER	EE XXXX	
		assoluto, X	INC OPER, X	FE XXXX	
INX incremento del registro indice X di un'unità	X←X+1	implicito	INX	E8	N Z
INY incremento del registro indice Y di un'unità	Y←Y+1	implicito	INY	C8	N Z
JMP salto incondizionale	OPER PC←(OPER)	assoluto	JMP OPER	4C XXXX	
		indiretto	JMP (OPER)	6C XXXX	
JSR chiamata di sottoprogramma: - rimemorizzazione di PC+2 (prossima istruzione al ritorno) - salto incondizionale	PC+2↓stack PC←OPER	assoluto assoluto	JSR OPER	20 XXXX	
LDA caricamento del contenuto dell'accumulatore	A←M	immediato	LDA # OPER	A9 XX	N Z
		pagina zero	LDA OPER	A5 XX	
		pagina zero, X	LDA OPER, X	B5 XX	
		assoluto	LDA OPER	AD XXXX	
		assoluto, X	LDA OPER, X	BD XXXX	
		assoluto, Y	LDA OPER, Y	B9 XXXX	
		(indiretto, X)	LDA (OPER, Y)	A1 XX	
		(indiretto), Y	LDA (OPER), Y	B1 XX	

(segue)

(seguito)

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
LDX caricamento del contenuto del registro di indice X	X←M	immediato pagina zero pagina zero, Y assoluto assoluto, Y	LDX # OPER LDX OPER LDX OPER, Y LDX OPER LDX OPER, Y	A2 A6 B6 AE BE	N Z
LDY caricamento del contenuto del registro di indice Y	Y←M	immediato pagina zero pagina zero, X assoluto assoluto, X	LDY # OPER LDY OPER LDY OPER, X LDY OPER LDY OPER, X	AO XX A4 XX B4 XX AC XXXX BC XXXX	N Z
LSR spostamento di una posizione verso destra del contenuto - dell'accumulatore - di una cella di memoria	0 → [→] → C	accumulatore pagina zero pagina zero, X assoluto assoluto, X	LSR A LSR OPER LSR OPER, X LSR OPER LSR OPER, X	4A XX 46 XX 56 XXXX 4E XXXX 5E XXXX	N=0 Z
NOP istruzione nulla (usata, ad esempio, per i ritardi)	vuoto	implicito	NOP	Ea	
ORA or logico dei contenuti dell'accumulatore e di una cella di memoria	A A+M V	immediato pagina zero pagina zero, X assoluto assoluto, X assoluto, Y (indiretto, X) (indiretto), Y	ORA OPER ORA OPER ORA OPER, X ORA OPER ORA OPER, X ORA OPER, Y ORA (OPER, X) ORA (OPER), 1	09 XX 05 XX 15 XX 0D XXXX 1D XXXX 19 XXXX 01 XX 11 XX	N Z
PHA memorizzazione dell'accumulatore nello stack	A↓ stack	implicito	PHA	48	
PHP memorizzazione del registro di stati P nello stack	P↓ stack	implicito	PHP	08	
PLA ripristino dell'accumulatore dello stack	A↑ stack	implicito	PLA	68	
PLP ripristino del registro di stati P dallo stack	P↑ stack	implicito	PLP	28	P

(segue)

(seguito)

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
ROL rotazione circolare di una posizione *via* il bit di riporto verso la sinistra del contenuto dell'accumulatore o di una cella di memoria		accumulatore pagina zero pagina zero, X assoluto assoluto, X	ROL A ROL OPER ROL OPER, X ROL OPER ROL OPER	2A 26 XX 36 XX 2E XXXX 3E XXXX	N Z C
ROR rotazione circolare di una posizione *via* il bit di riporto verso la destra del contenuto dell'accumulatore o di una cella di memoria		accumulatore pagina zero pagina zero, X assoluto assoluto, X	ROR A ROR OPER ROR OPER, X ROR OPER ROR OPER, X	6A 66 XX 76 XX 6E XXXX 7E XXXX	N Z C
RTI ritorno d'interruzione	P↑PC↑	implicito	RTI	40	P ripristinato
RTS ritorno dalla subroutine	PC↑ PC←PC+1	implicito	RTS	60	
SBC sottrazione del contenuto dell'accumulatore del contenuto di una cella di memoria e del bit di riporto	A←A-M-C	immediato pagina zero pagina zero, X assoluto assoluto, X assoluto, Y (indiretto, X) (indiretto), Y	SBC OPER SBC OPER SBC OPER, X SBC OPER SBC OPER, X SBC OPER, Y SBC (OPER, X) SBC (OPER), Y	E9 XX E9 XX F5 XX ED XXXX FD XXXX F9 XXXX E1 XX F1 XX	
SEC posizionamento a 1 del bit di riporto	C←1	implitico	SEC	38	C=1
SED passaggio in modo decimale	D 1	implicito	SED	F8	C=1
SEI divieto delle interruzioni	I←I	implicito	SEI	78	I=1

(segue)

(seguito)

Mnemonico e descrizione	Operazione	Modo d'indirizzamento	Forma assembleare	Codice operazione	Flags modificati
STA memorizzazione del contenuto dell'accumulatore in una casella di memoria	M←A	pagina zero pagina zero, X assoluto assoluto, X assoluto, Y (indiretto, X) (indiretto), Y	STA OPER STA OPER, X STA OPER STA OPER, X STA OPER, Y STA (OPER, X) STA (OPER), Y	85 XX 95 XX 8D XXXX 9D XXXX 89 XXXX 81 XX 91 XX	
STX memorizzazione del contenuto del registro indice X in una cella di memoria	M←X	pagina zero pagina zero, Y assoluto	STX OPER STX OPER, Y STX OPER	86 XX 96 XX 8E XXXX	
STY memorizzazione del contenuto del registro indice X in una cella di memoria	M←Y	pagina zero pagina zero pagina zero	STY OPER STY OPER, X STY OPER	84 XX 94 XX 8C XXXX	
TAX caricamento nel registro di indice X del contenuto dell'accumulatore	X←A	implicito	TAX	AA	M Z
TAY caricamento nel registro Y del contenuto dell'accumulatore	Y←A	implicito	TAY	A8	N Z
TSX caricamento nel registro di indice X del puntatore di stack S	X←S	implicito	TSX	BA	N Z
TXA trasferimento nell'accumulatore del registro di indice X	A←X	implicito	TXA	8A	N Z
TXS trasferimento nello stack pointer S del registro di indice X	S←X	implicito	TXS	9A	
TYA trasferimento nell'accumulatore del contenuto del registro di indice Y	A←Y	implicito	TYA	98	N Z

4.2.4 Tecniche di base di programmazione in assemblatore 6502

Il lettore che conosce solamente il Basic deve conoscere le metodologie relative al linguaggio assembler che qui di seguito descriveremo, affinché lo possa utilizzare facilmente:

- **Istruzioni di salto**

Istruzioni di salto incondizionato. L'istruzione GOTO numero riga del Basic viene sostituita in assembler dall'istruzione JMP indirizzo.

Istruzioni di salto condizionato. Per sostituire l'istruzione Basic IF...GOTO, dovrete eseguire le seguenti due fasi:

- eseguire il confronto per determinare qual è la condizione che deve essere testata sui bits del registro di indicatore di stato. Questo avviene mediante una di queste tre istruzioni: CMP, CPX, CPY;
- testare la condizione desiderata ed effettuare il salto tramite le istruzioni BEQ, BMI, BPL, BNE, BCC, BCS, BVC, BVS se essa è verificata.

Esempio:

```
 10 A=PEEK($1000)
 20 IF A=PEEK($100) GOTO 100
100 END
```

può essere scritto in assembler in questo modo:

```
$800 LDA $1000
     CMP $100
     BEQ $10     ┐
     trattamento │  salto di $10 bytes
     RTS        <┘
```

Per = adopereremo BEQ.
Per < > adopereremo BNE.
Per il < adopereremo BMI.
Per il > adopereremo BPL.

Le altre istruzioni (BCC, BCS, BVC, BVS) sono legate all'aritmetica binaria.

- **Operazioni logiche**

- Somma logica (ORA).
- Prodotto logico (AND).
- Or esclusivo logico (EOR).

Queste vengono eseguite tra il contenuto dell'accumulatore e il contenuto di una cella di memoria oppure con un valore numerico predefinito.

OR

Bit M \ Bit ACC	0	1
0	0	1
1	1	1

AND

Bit M \ Bit A	0	1
0	0	0
1	0	1

OR ESCLUSIVO

Bit M \ Bit A	0	1
0	0	1
1	1	0

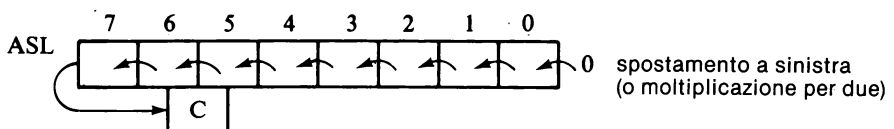
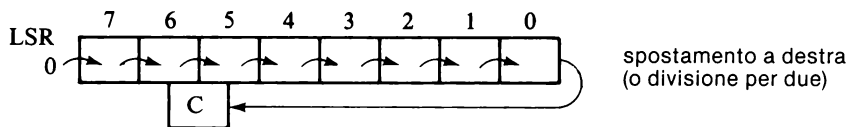
Il loro risultato si trova nell'accumulatore mentre i bits relativi dell'indicatore vengono posizionati. Risulta quindi inutile l'indirizzo di una CMP dopo AND, OR, EOR.

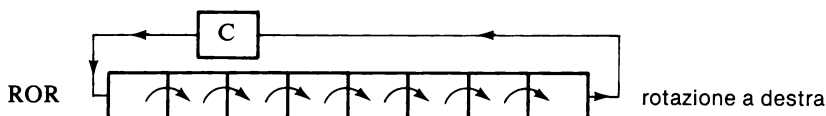
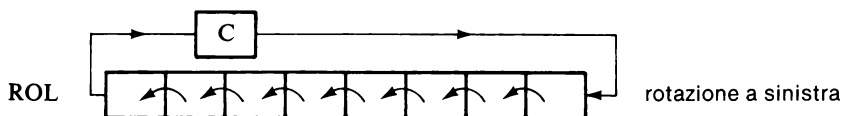
```
10 A=PEEK(1000)
20 IF (A MOD 256)=0 GOTO 100
```

può essere scritto in assembler:

```
LDA $1000
AND $OF
BEQ indirizzo voluto
```

- **Operazioni di Shift e di rotazione**

Shift. Un'operazione di Shift permette di spostare il contenuto dell'accumulatore (oppure di una cella di memoria) di un bit verso sinistra oppure verso destra.
Il bit che fuoriesce dal registro si trova nel bit di riporto; il bit all'altra estremità assume il valore 0.

Il microprocessore 6502 non è ricco di operazioni di Shift. Esso difetta ad esempio degli Shift aritmetici che conservano il bit 7 (bit di segno) che vengono utilizzati soprattutto nell'aritmetica in virgola mobile.

Rotazioni. Una rotazione consente lo spostamento del contenuto dell'accumulatore (o di una cella di memoria) di un bit verso sinistra oppure verso destra con la differenza rispetto allo Shift che il bit fuoriuscito viene inserito all'estremità opposta del registro.

Il microprocessore 6502 non è altresì ricco in rotazioni. Esso non ammette delle rotazioni su 8 bits, dove quello che esce da un lato entra dall'altro.

• Operazioni aritmetiche

Le operazioni aritmetiche del 6502 sono in numero di due. Esse sono le seguenti:

- ADC somma con riporto;
- SBC sottrazione con riporto.

Non esiste la somma senza riporto nel 6502. Questo è un difetto perché è necessario eseguire un'istruzione CLC prima di ogni operazione di somma. Inoltre non esistono istruzioni di somma né di sottrazione su 16 bits come per la famiglia INTEL ZILOG. È possibile però simularle eseguendo due successive operazioni di somma su 8 bits conservando il bit di riporto.

Esempio:

OP1 all'indirizzo AD1
OP2 all'indirizzo AD2
RES all'indirizzo ADR

Il programma di somma su 16 bits è il seguente:

```
CLC
LDA AD1     )
ADC AD2     } parte meno significativa
STA ADR     )

LDA AD1+1   )
ADC AD2+1   } parte più significativa
STA ADR+1   )
```

Lo stesso principio viene adottato per eseguire la sottrazione a 16 bits:

```
SEC
LDA AD1     )
SBC AD2     } parte meno significativa
STA ADR     )

LDA AD1+1   )
SBC AD2+1   } parte più significativa
STA ADR+1   )
```

Il microprocessore 6502 consente di eseguire le operazioni aritmetiche sia in modo esadecimale, sia in DCB (decimale). La modalità di funzionamento è diversa da quella delle famiglie INTEL ZILOG.
Per porsi nella modalità esadecimale, dovete utilizzare l'istruzione CLD che forza il bit D a 0. Per la modalità BCD, bisognerà utilizzare l'istruzione SED.
Non esiste l'istruzione di moltiplicazione né di divisione. Questo non è un handicap del 6502 perché tutti i microprocessori su 8 bits non consentono tali operazioni. Ecco ad esempio un programma che consente di effettuare il prodotto fra due bytes in memoria posti agli indirizzi AD1 e AD2 e che fornisce il risultato in AD3.

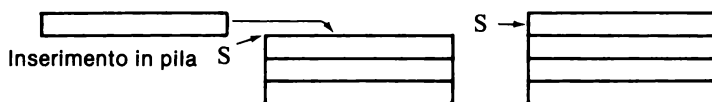
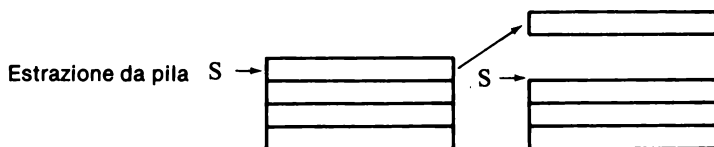
```
MULTIP  LDA #0      )
        STA AD3     } Inizializzazione del risultato a 0
        LDX # 8     } X è un contatore di Shift
BOUC    LSR AD2       Shift del moltiplicatore
        BCC VIDE      Se il bit è forzato a 0 nessuna somma
        CLC         ) Se il bit è uguale a 1 somma il moltipli-
        ADC AD2     ) cando al risultato
VIDE    ROR A
        ROR AD3
        DEX         )
        BNE BOUC    } È l'ultimo Shift?
```

Lo stesso principio può essere utilizzato su 16 bits. È necessario però in più gestire gli Shifts dei 16 bits tra i due bytes.

• **Sottoprogramma**

L'istruzione GOSUB del Basic è sostituita dalla CALL indirizzo.
L'istruzione RETURN del Basic è sostituita dalla RTS indirizzo.
L'indirizzo di ritorno del sottoprogramma è conservato nella pila. Questo implica che ogni sottoprogramma deve lasciare pulita la pila al ritorno (dobbiamo ritrovare la stessa pila sia al ritorno che alla chiamata).

• **Utilizzo della pila**

Questa è una facility offerta dal programma assemblatore per immagazzinare delle variabili locali per un breve periodo; infatti la pila è quasi indispensabile per avere dei sottoprogrammi. Tutti i microprocessori più celebri a 8 bits (Motorola 600B, INTEL 8080 e 8085, ZILOG Z80) possiedono una pila. Sarebbe un peccato non utilizzarla.
Due istruzioni di messa in pila sono disponibili: PHA e PHP.

Esistono inoltre due istruzioni di estrazione da pila a disposizione: PLA e PLP.

• **Operazioni di ingresso-uscita**

Le istruzioni di ingresso-uscita sono delle istruzioni specializzate per la gestione delle unità di ingresso-uscita. Il microprocessore 6502 utilizza dei dispositivi ingresso-uscita realizzati in memoria, il che significa che i chips di interfaccia sono collegati al bus indirizzi come una memoria. Essi sono visti dal programmatore come dei veri e propri indirizzi di memoria.
Questo però rallenta il trattamento delle operazioni ingresso-uscita ed è utile a questo scopo il poter disporre di un meccanismo rapido con i chips che risiedono in pagina zero. Tuttavia questa è in genere impiegata dalla RAM: è questo il caso dell'Apple II.

4.2.5 Consigli al lettore principiante

Vi consigliamo di procedere nella maniera seguente:

- comprendere le diverse istruzioni e la struttura interna del 6502;
- eseguire dei piccoli esercizi con il miniassemblatore o con il LISA;
- utilizzare l'assemblatore per la creazione di sottoprogrammi richiamabili dal Basic sfruttando al massimo il Monitor (cfr. par. 4.3.4);
- studiare molto approfonditamente il Monitor essendo esso molto efficiente rispetto allo spazio occupato.

4.3 IL MONITOR

4.3.1 Presentazione

Il Monitor è un programma residente in ROM che consente le operazioni di ingresso-uscita e offre all'utilizzatore dei comandi che permettono la messa a punto dei programmi in linguaggio macchina. Studieremo in questo paragrafo i seguenti argomenti:

- accesso al Monitor dal Basic;
- comandi del Monitor (esame dei registri, della memorià, ecc.);
- miniassemblatore e messa a punto dei programmi;
- mappa di memoria dell'Apple e struttura interna del Monitor.

4.3.2 Come si accede al Monitor dal Basic

• Accesso al Monitor

Esistono due versioni del Monitor: la versione standard (Apple II) e la versione Autostart (Apple II+).
Se avete un Apple II standard, accedete al Monitor quando mettete in moto il sistema. In questo caso vedrete in basso e a sinistra dello schermo una stessa seguita da un cursore lampeggiante: il Monitor aspetta il vostro comando.
Se avete invece un Apple II+ o se volete ritornare al Monitor partendo dall'Integer Basic o dall'Applesoft, dovrete digitare il seguente comando:

> oppure] CALL-151

Questo comando corrisponde alla chiamata del sottoprogramma posto all'indirizzo $FF69. Quando l'asterisco viene messo in evidenza vi troverete nel Monitor, il quale sarà in attesa di un vostro comando.

• Uscita dal Monitor

Esistono molte possibilità di uscire dal Monitor per ritornare al Basic; esse sono le seguenti:

1. digitare CTRL-C↵ il che permette di passare al Basic in ROM salvando il programma e le sue variabili;
2. digitare CTRL-B↵ il che permette di passare al Basic in ROM distruggendo il programma e le sue variabili;

3. digitare 3DOG↙ il che permette di passare al Basic caricato in memoria dal dischetto salvando il programma e le sue variabili (è l'equivalente del CTRL-C per il passaggio al Basic non posto in ROM);
4. digitare OG↙ il che permette la stessa cosa del punto 3 partendo da una cassetta.

*Le ultime due possibilità sono utilizzate quando si lavora con l'*Applesoft *su un* Apple II *standard.,*
Esempi di utilizzo delle possibilità di accesso e di uscita dal Monitor su un Apple II+ (Applesoft in ROM).

```
*CTRL-B
]LIST            } uscita dal Monitor con distruzione delle variabili
]PRINT V

o
]10V=110
]20PRINT V
]LIST
10V=110          } ingresso ed esecuzione del programma Basic
20PRINT V
30RUN
110

]CALL-151
*CTRL-C
]LIST
10V=110          } passaggio al Monitor e ritorno tramite CTRL-C:
20PRINT V          il programma e i dati sono intatti
]PRINT V
110
]
```

4.3.3 I comandi del Monitor

I comandi del Monitor offrono all'utente delle facilitazioni per la messa a punto dei programmi in linguaggio macchina, come le seguenti:

- modifica della memoria, copia di una zona di memoria, salvataggio di una zona di memoria su cassetta e caricamento a partire dalla cassetta;
- creazione di comandi utente;
- disassemblaggio di una zona di memoria, esame dei registri ecc.

Il Monitor riconosce 22 caratteri come comandi propri. Esso interpreta solamente il primo carattere di ogni comando. Gli indirizzi e i dati devono essere espressi in forma esadecimale (0...9ABCDEF). Gli indirizzi sono rappresentati tramite 4 cifre esadecimali, mentre i dati da 2. Il Monitor

completa i valori forniti inserendo degli zeri a sinistra per arrivare al numero di cifre esadecimali necessarie, oppure esegue un troncamento a sinistra se è stato fornito un numero di valori oltre il necessario.

• **Esame e modifica della memoria**

Esame della memoria. È possibile esaminare la memoria in due modi:

- sia per byte singolo, sia per blocchi di otto bytes alla volta (questa procedura verrà chiamata da noi "passo passo");
- sia tramite i bytes di una zona di memoria compresa tra due indirizzi.

* ESAME PASSO PASSO

Per esaminare il contenuto di un byte che si trova ad un dato indirizzo, è sufficiente digitare questo indirizzo: il Monitor allora vi segnalerà il valore del byte.

Per esempio, se digitate *ABCD
il Monitor risponderà ABCD-D7 = contenuto del byte di indirizzo ABCD
*

Il Monitor ha dunque inizializzato un puntatore in memoria al valore ABCD. Per esaminare i bytes successivi, è sufficiente digitare il comando RETURN. Il Monitor evidenzierà allora il contenuto del byte successivo fino alla fine del blocco di otto bytes nel quale ci si trova.

```
Per esempio se digitate   *ABCD
                           ABCD-D7
in seguito                *RETURN
il Monitor segnalerà      A5D6
                          *
```

Se ridigitate RETURN allorché il puntatore si trova ad un inizio di un blocco di 8 bytes, esso segnalerà l'indirizzo di inizio del blocco e i valori degli 8 bytes.

Nell'esempio che segue, se digitate ancora RETURN, il Monitor evidenzierà:

```
ABD0- 30 40 50 60 20 10 00 70
*
```

Per continuare l'esposizione dei blocchi successivi vi basterà digitare RETURN tutte le volte che sarà necessario.

* ESAME DI UNA ZONA DI MEMORIA

È possibile esaminare una zona di memoria di dimensione superiore a quella di un blocco di otto bytes. Per ottenere ciò basterà:

- fornire gli indirizzi di inizio e fine della zona;
- fornire l'indirizzo di fine della zona, essendo quella di inizio per convenzione presa uguale a quella del puntatore.

Se digitate ad esempio: *E015.E025
il Monitor risponderà:

E015-4C ED FD fine del blocco corrente di otto bytes
E018-A9 2O C5 24 B0 0C A9 8D
E20-A0 07 20 ED FD A9 indirizzo di fine blocco E025
*

Il Monitor quindi evidenzia:

- l'indirizzo di partenza seguito dal numero di bytes necessari per arrivare al prossimo blocco di otto bytes;
- tutti i blocchi di otto bytes nei quali non si trova l'indirizzo di fine zona;
- l'indirizzo di inizio dell'ultimo blocco e il numero di bytes necessari per arrivare all'indirizzo della fine della zona.

Se volete continuare ad esaminare il seguito della memoria, sarà sufficiente digitare

.indirizzo di fine.

Nell'esempio che riportiamo di seguito:

se digitate: *.E028
il Monitor evidenzierà: E026 AA AB AC
*

È spesso indispensabile interrompere la visualizzazione sullo schermo. Per fare ciò è sufficiente digitare CTRL-S. *Per riprendere la visualizzazione, vi sarà sufficiente digitare uno spazio.*

Modifica della memoria. È possibile modificare la memoria in due modi:

- byte per byte;
- più bytes consecutivi alla volta in memoria.

* MODIFICA DI UN BYTE PASSO PASSO

Se volete modificare il contenuto di un indirizzo di memoria, bisognerà innanzitutto inizializzare il puntatore di memoria e quindi modificare il byte posto a quest'indirizzo.
Supponiamo ad esempio di voler inserire il valore AA all'indirizzo 0: dovrete digitare

*0 indirizzo da inizializzare
0000-XX
*:AA valore da inserire

Per verificare digitate 0. Il Monitor segnalerà il valore e potrete constatare che il valore è stato veramente modificato.

*0
0000-AA
*

Se adesso volete modificare il valore contenuto all'indirizzo successivo (nel caso precedente esso è uguale a uno), vi sarà sufficiente digitare il valore da

inserire (il puntatore già inizializzato a 0 è stato incrementato di 1 quando abbiamo inserito AA all'indirizzo 0; esso quindi vale 1).

*:AB

Ancora una volta il puntatore è stato incrementato e quindi per inserire dei valori nelle celle di memoria successive, sarà sufficiente digitarli, ciascuno di essi preceduto da :.
Potete eseguire le due operazioni di inizializzazione del puntatore e di modifica del valore del byte, mediante un solo comando.
Nell'esempio seguente, invece di digitare i due comandi:

*0
e *:AA

potevate semplicemente digitare il comando seguente:

*0:AA

Il risultato ottenuto sarebbe stato il medesimo.

* MODIFICA DI PIÙ BYTES CONSECUTIVI IN MEMORIA

È alquanto fastidioso dover digitare ad ogni byte:

: valore del byte

Il Monitor vi consente di digitare più bytes alla volta, l'uno dopo l'altro.
Ad esempio se digitate:

*3000:00 01 02 03 04 05 06 07 08 09 0A

otterrete in memoria dopo l'esecuzione del comando:

```
3000 00
3001 01
3002 02
3003 03
3004 04
3005 05
3006 06
3007 07
3008 08
3009 09
300A 0A
```

Volete verificare! Digitate il comando seguente:

*3000.300A

Il Monitor vi risponderà:

```
3000-00 01 02 03 04 05 06 07
3008-08 09 0A
*
```

Il Monitor accetta fino ad 83 valori se il puntatore è inizializzato e fino ad 84 se non lo è. Tuttavia vi sconsigliamo fortemente di digitare tanti valori essendo elevatissimo il rischio di errore.

• Ricopiatura e confronto di zone di memoria

È possibile il trattamento delle zone di memoria come se esse fossero delle entità proprie. Il Monitor vi consente la ricopiatura di una zona di memoria e il confronto di due zone.

Ricopiatura di una zona di memoria. Per ricopiare una zona di memoria il Monitor deve conoscere i seguenti parametri:

- indirizzi d'inizio e di fine della zona di memoria da ricopiare;
- indirizzo di inizio della zona di memoria dove va ricopiata la zona precedente.

Il formato del comando è il seguente:

{indirizzo ricevente} < {indirizzo di inizio} . {indirizzo di fine} Move

Ad esempio se digitate il comando seguente:

*10<0.F Move

il Monitor eseguirà la ricopiatura dei 16 bytes posti agli indirizzi da 0 a F partendo dall'indirizzo /0.
Se digitate i comandi

*0.F

e

*10.1F

otterrete lo stesso risultato.
Quando il Monitor esegue la ricopiatura della zona, esso non modifica in alcun modo la zona di memoria che esso ricopia. Se l'indirizzo di fine zona da ricopiare è inferiore o uguale all'indirizzo di inizio, il Monitor eseguirà la ricopiatura del primo byte della zona.
Se l'indirizzo ricevente è compreso tra l'indirizzo di inizio della zona da ricopiare e l'indirizzo di fine di questa zona, l'inizio della zona sarà ricopiato un certo numero di volte distruggendo il resto della zona (cfr. par. 4.3.3).

Confronto di due zone di memoria. Potete utilizzare il Monitor per confrontare due zone di memoria. Questa possibilità è utile quando ad esempio si salva una zona di memoria su una cassetta e si vuol verificare che il salvataggio è stato eseguito correttamente (cfr. par. 4.3.3).
Il formato di questo comando è identico al precedente:

{indirizzo ricevente della zona riferita} < {indirizzo di inizio della zona da confrontare} . {ind. di fine della zona da confrontare} Verify

Per esempio, se volete digitare il comando seguente:

*10<0.F Verify

dopo aver digitato

*10<0.F Move

Il Monitor deve passarvi la mano senza segnalare alcunché se la ricopiatura è stata eseguita correttamente.
Per contro, se digitate i comandi seguenti:

*00:0A
*10<0.FM
*1A:1A
*10<0.FV

il Monitor segnalerà sullo schermo

00A-0A (1A)

Per ogni differenza riscontrata durante il confronto, il Monitor segnala:

- l'indirizzo posto tra gli indirizzi di inizio e di fine della zona da confrontare;
- il valore trovato a quest'indirizzo e, fra parentesi, il valore trovato in corrispondenza all'indirizzo nell'altra zona.

Dovete sempre mettere l'indirizzo della zona della quale siete sicuri a sinistra del segno <. In questo modo potrete conoscere gli indirizzi in cui sono localizzati gli errori e non quelli in cui si trovano i valori corretti.

Inizializzazione di una zona di memoria. Il comando Move consente di inizializzare una zona di memoria corrisponde a ricopiare un byte tante volte quanto è necessario.
Supponiamo di voler inizializzare la zona di memoria compresa tra gli indirizzi 0 e FF. Ad esempio l'esecuzione dei comandi seguenti consente di fare ciò:

*0:0
*1<0.FEM

Il primo comando inizializza il byte di indirizzo 0 al valore 0 (cfr. par. 4.3.3).
Il secondo comando ricopia il valore del byte di indirizzo 0 a quello di indirizzo 1. Il byte di indirizzo 1 viene posto quindi al valore 0. Il byte di indirizzo 1 viene ricopiato all'indirizzo 2 e così via. Tutti i bytes di indirizzo compreso tra 0 e FF sono quindi inizializzati a 0.
È possibile riprodurre nello stesso modo delle strutture di dimensione superiore ad 1 byte.
Supponiamo di voler inizializzare la zona compresa tra gli indirizzi 0 e 256 nel modo seguente:

0 1 2 3 4 5 6 7 0...7 0...7 0...7 0...7 0...7

cioè (0...7) tante volte quanto è necessario. Per questo digitiamo i seguenti comandi

*0:0 1 2 3 4 5 6 7
*8<0.F7M

Dopo le prime 8 ricopiature, la struttura è stata ricopiata dagli indirizzi 8 a F. Dopo le 8 ricopiature successive la struttura è stata ricopiata dagli indirizzi 10 a 17 e così di seguito.
L'indirizzo ricevente è uguale all'indirizzo di inizio della zona da inizializzare sommata alla lunghezza della struttura. L'indirizzo di fine di tale zona è quello della zona da inizializzare meno la lunghezza della struttura.
In genere, per inizializzare una zona con una struttura di lunghezza n, risulterà necessario:

- inizializzare i primi n bytes della zona della struttura;
- digitare il seguente comando

{inizio zona + n} < {inizio zona} {fine zona–n} M

È possibile, seguendo lo stesso principio, verificare che una zona è stata inizializzata correttamente utilizzando il comando Verify.

{inizio zona + n} < {inizio zona} {fine zona–n} V

• **Salvataggio e ricaricamento di una zona di memoria**

Il Monitor possiede due comandi speciali che gli consentono il salvataggio di una zona di memoria su cassetta e quindi di ricaricarla per un successivo utilizzo. Questa zona di memoria può contenere ad esempio un sottoprogramma in linguaggio macchina 6502 oppure una struttura grafica ad alta risoluzione. In ogni caso per i possessori di DOS è preferibile il suo utilizzo per ragioni di velocità e di sicurezza.

Salvataggio di una zona di memoria su cassetta. Il comando WRITE di salvataggio di una zona di memoria su cassetta ha il formato seguente:

{indirizzo d'inizio} . {indirizzo di fine} W

Ad esempio il comando

*0.FFFF W

vi consentirà il salvataggio di 64 K bytes su una cassetta.
Per effettuare un salvataggio dovete rispettare il seguente procedimento:

- predisporre il registratore nella posizione di registrazione;
- lasciare girare il nastro per qualche secondo;
- digitare il comando.

Il Monitor scriverà allora una testata la cui durata è di 10 secondi sulla cassetta e in seguito i dati. Alla fine scriverà un byte il cui valore è uguale alla somma di tutti i bytes registrati troncata a 8 bits.
Dopo aver terminato emetterà un suono e passerà la mano all'utilizzatore.
Il Monitor invia all'incirca 1350 bits al secondo sulla cassetta, il che equivale a 4 K bytes in 35 secondi.
Il Monitor non sa se il registratore è pronto e nemmeno se esso è collegato. È l'utilizzatore che deve verificare se tutto è predisposto per il registratore.

Ricaricamento di una zona di memoria partendo dalla cassetta. Il comando READ consente di caricare i dati registrati su una cassetta trasferendoli in memoria. La zona di ricarico non è necessariamente situata allo stesso indirizzo di quella che è stata oggetto di salvataggio. Il formato del comando Read è il seguente: dovete fornire al Monitor l'indirizzo in cui deve essere ricaricato il primo byte letto sulla cassetta e l'indirizzo in cui deve essere ricaricato l'ultimo letto. Questi due indirizzi devono essere separati da un punto come indica il comando seguente:

{indirizzo di inizio} . {indirizzo di fine} R

Per leggere una cassetta dovete eseguire le seguenti operazioni:

1. Porre la cassetta dopo l'inizio dell'intestazione di durata di 10 secondi inserita dal Monitor prima dei dati: per fare ciò sarà sufficiente ascoltare la cassetta; infatti l'intestazione è l'unica parte della cassetta che non provoca alcun rumore a differenza dei dati.
2. Digitare il comando: il Monitor si mette in attesa delle informazioni che gli arrivano dal dischetto; gli saranno necessari all'incirca 3 secondi di intestazione per sincronizzarsi sulla cassetta.
3. Premere il tasto PLAY: il Monitor caricherà allora i dati in memoria. Se esso riscontrerà un errore emetterà la segnalazione ERR; altrimenti nulla verrà segnalato. Al termine il Monitor emetterà un "beep" all'altoparlante e passerà la mano all'utente.

*Il Monitor non sa se il registratore è predisposto e neppure se ne esiste uno collegato all'*Apple. *È l'utente stesso che deve verificare che tutto è pronto per il registratore. Il Monitor rimane bloccato finché non si è sincronizzato sulla cassetta.*
Se digitate ad esempio il comando:

*200.300 R

il Monitor caricherà i 257 bytes dalla cassetta.
Quali sono le fonti di errore durante la lettura di una cassetta?

- lunghezza dell'intestazione inferiore a 3 secondi;
- lunghezza della parte da ricopiare diversa dalla lunghezza salvata sulla cassetta;
- cassetta cancellata oppure difettosa;
- errore durante la scrittura della cassetta.

Come vengono individuati i due primi tipi di errore? L'ultimo byte registrato dal Monitor è stato calcolato in base ai bytes precedentemente registrati; quindi in caso di errore, il byte calcolato dopo la lettura dei bytes sulla cassetta, sarà diverso da esso tranne che per pura coincidenza.
Nel primo caso potete ascoltare la cassetta. Se riscontrate che l'intestazione è effettivamente di durata inferiore a 3 secondi, vi sarà necessario rifare la cassetta (o solamente una parte di essa se la cassetta deve essere utilizzata parecchie volte). Altrimenti riprovate. Il caricamento dovrebbe effettuarsi correttamente.
Per evitare che si verifichi il secondo caso vi consigliamo di segnare sulla cassetta la registrazione che vi sono state eseguite.
Il miglior modo per non avere dei problemi, è quello di provare a leggere la cassetta finché la zona di memoria che è stata salvata non sia stata ancora cancellata (cfr. 4.3.3). Così potrete rifare la manipolazione tutte le volte che risulterà necessario.

Salvataggio di una zona di memoria su dischetto. Se avete un'unità per dischetti, eseguire il salvataggio di una zona di memoria è molto più rapido e facile che mediante una cassetta.
Per eseguire il salvataggio di una zona di memoria, dovete uscire dal Monitor ed entrare nel Basic, avendo il DOS in memoria (digitate CTRL–C).

Esempio: per eseguire il salvataggio di una zona di memoria di indirizzo $200, di lunghezza 200, nel file SPMACH (dischetto di volume 254, drive 1 dello slot 6), dovete digitare il seguente comando:

BSAVE SPMACH, A$200, L200, S6, D1, V254

La lunghezza massima che può essere oggetto di salvataggio ha una lunghezza di 32 K bytes. Il comando BSAVE vi consente di digitare le costanti in base decimale o esadecimale. In quest'ultimo caso non dimenticate di far precedere le costanti dal segno $.
Gli ultimi tre parametri sono facoltativi.

Ricaricamento di una zona di memoria da un dischetto. Per caricare in memoria il contenuto di un file, dovete utilizzare il comando BLOAD del DOS.
Ad esempio se digitate:

BLOAD SPMACH, A$290, L200, S6, D1, V254

il DOS caricherà all'indirizzo $290 il file SPMACH.
L'indirizzo presente nel comando è quello al quale deve essere caricato il file. Se quest'indirizzo non è presente nel comando, il file viene caricato all'indirizzo presente nella BSAVE. La lunghezza è un parametro opzionale, poiché il DOS interrompe la ricopiatura del file quando ne incontra la fine.
Quando caricate un file in memoria, esso ne distrugge una zona. Fate

attenzione a non distruggere il DOS, *le pagine grafiche, il programma e le variabili Basic. Inoltre fate attenzione a non distruggere l'interprete Basic se esso non si trova su* ROM *(esempio tipico Applesoft caricato da un dischetto).*

Verifica della zona di memoria salvata su una periferica. Per essere sicuri del salvataggio testé eseguito, esiste un metodo che consente di rileggere immediatamente i dati salvati, caricandoli in memoria a un indirizzo diverso dall'indirizzo della zona che è stata salvata e infine confrontando in memoria le due zone mediante il comando VERIFY del Monitor.
Vi forniamo qui due esempi di applicazione che possono essere utilizzati per un sottoprogramma assembler, una struttura grafica o qualunque altro dato.

* CASSETTA

Se utilizzate le cassette per il salvataggio delle zone di memoria, la prima fase è quella di salvare la zona di memoria con il comando Write.
Digitate ad esempio:

*500.5FF W

Il Monitor salva i bytes posti tra gli indirizzi $500 e 45FF sulla cassetta. Quando ha finito, l'altoparlante emetterà un "beep". Dovete allora riavvolgere il nastro fino all'inizio dell'intestazione sonora e digitate un comando che incarichi il Monitor di caricare la cassetta in memoria. Supponiamo ad esempio che la zona di memoria 400.4FF sia libera. Potete allora digitare il comando seguente:

*$400.4FF R

Avremo allora in memoria:

- tra gli indirizzi $400 e $4FF i dati letti dalla cassetta;
- tra gli indirizzi $500 e $5FF i dati di riferimento.

A questo punto confronterete le due zone, digitando il comando:

*500<400.4FF V

Il Monitor eseguirà il confronto fra le due zone. Se non riscontrerà degli errori, potrete considerare buona la cassetta, altrimenti dovrete ricominciare il salvataggio.

* DISCHETTO

Il principio è lo stesso per i dischetti per i quali bisogna avere il DOS in memoria ed essere all'interno del Basic.
Digitate allora questi comandi:

BSAVE MEMORIA, A$500, L$FF	salvataggio della zona di memoria
BLOAD MEMORIA, A$400	ricarico della zona ad un altro ind.

```
CALL-151                      passaggio al Monitor
*500<400.4FF V                confronto delle due zone
```

Se il Monitor non riscontra alcuna differenza, potete allora essere sicuri che il file è l'esatta rappresentazione della zona di memoria precedentemente salvata. Altrimenti ritornate al Basic (CTRL-C oppure 3DOG) e rieseguite il salvataggio.

• Messa a punto di programmi in linguaggio macchina e in assembler

Il Monitor offre all'utilizzatore una consistente assistenza per la messa a punto di sottoprogrammi in assembler e in linguaggio macchina. Tuttavia alcune di queste possibilità offerte (miniassemblatore, possibilità d'esecuzione passo passo...) non sono disponibili che con la versione standard del Monitor (Apple II standard) e non sono accessibili con la versione ROM autostart per motivi di occupazione di memoria. In questo paragrafo studieremo ciascuna di queste possibilità precisando se esse sono disponibili o non con la ROM autostart (Apple II+).

Esame e modifica dei registri del microprocessore 6502. Il Monitor riserva 5 indirizzi in memoria per salvare i 5 registri del microprocessore 6502: A, X, Y, P (stato del microprocessore), S (puntatore di pila). Il comando Examine, che viene eseguito dal Monitor digitando CTRL-E, ordina al Monitor stesso di visualizzare il contenuto di questi indirizzi.
Se digitate il seguente comando:

```
*CTRL-E
```

il Monitor visualizzerà il contenuto dei registri nel seguente formato:

```
A=XX  X=XX  Y=XX  P=XX  S=XX
```

Il Monitor vi permette di modificare il contenuto dei 5 registri del 6502. Digitate CTRL-E: il Monitor visualizza il contenuto dei registri e posiziona il puntatore di memoria sul primo di questi indirizzi. Dovete dunque solamente modificare il contenuto di questi indirizzi.
Supponiamo che il contenuto dei registri sia rispettivamente $BO $FF $FE/ $30 $F8 e che voi vogliate porre l'accumulatore A al valore $FF e il registro indice Y al valore $AA. La procedura da seguire è la seguente:

```
*CTRL-E
A=BO  X=FF  Y=FE  P=30  S=F8
*:FF  FF  AA
```

verificate

```
*CTRL-E
A=FF  X=XX  Y=AA  P=30  S=F8
```

Il Monitor caricherà i bytes relativi a questi indirizzi nei registri corrispondenti prima di ripassare la mano al programma utente che volete eseguire o mettere a punto (comandi G, S, T spiegati di seguito).

Se volete solo modificare il puntatore di pila, dovete ugualmente ripetere i valori dei registri precedenti (nella lista). In effetti, quando digitate CTRL-E, *il Monitor inizializza il puntatore di memoria all'indirizzo corrispondente all'accumulatore. Per accedere al puntatore di pila, che è il quinto indirizzo, bisogna modificare il puntatore di memoria modificando gli indirizzi precedenti.*
Per esempio, se volete porre il valore 00 *nel puntatore di pila, dovete digitare i comandi:*

```
*CTRL-E
A=FF     X=FF     Y=AA     P=30     S=F8
*:FF       FF       AA       30       00
valore   valore   valore   valore   nuovo valore
di A     di X     di Y     di P     di S
```

Creazione ed esecuzione di un programma in linguaggio macchina. È facile inserire in memoria un programma in linguaggio macchina. Preparate i codici da porre in memoria. Caricate quindi i codici in memoria utilizzando i comandi di modifica della memoria stessa. Vi consigliamo vivamente di salvare in seguito su cassetta o dischetto la zona di memoria in cui il programma viene caricato.
Ora voi volete provare il vostro programma. Supporremo, nelle spiegazioni che seguono, che il vostro programma inizi all'indirizzo $8000. Per lanciare il vostro programma digitate il comando:

```
*8000G
```

Il Monitor ricarica allora i registri a partire dagli indirizzi in cui essi sono normalmente salvati. Se per esempio avete modificato i registri prima di iniziare l'esecuzione del vostro programma, è in questo preciso momento che essi sono ricaricati. Il Monitor richiama in seguito il vostro programma, che considera come un sottoprogramma, utilizzando l'istruzione JSR. Il vostro programma prende a questo punto il controllo del microprocessore e viene eseguito.
Affinché il Monitor riprenda il controllo dopo l'esecuzione del vostro programma, l'ultima istruzione deve essere RTS (ritorno all'inizio di un sottoprogramma del programma principale). Altrimenti dovrete reinizializzare il vostro Apple.
Il formato abituale dell'istruzione GO è il seguente:

```
{indirizzo di inizio} G
```

Ecco due esempi riguardanti il comando GO:

Esempio 1. Se avete un Apple II standard e avete caricato l'Applesoft da un dischetto, per ritornare all'Applesoft a partire dal Monitor dovete digitare il comando 3DOG.
Questo comando corrisponde ad un salto all'indirizzo $3DO dove si trova l'istruzione JMP (salto) al DOS.

Esempio 2. Se digitate i comandi seguenti:

(input delle istruzioni)

```
*300:A9 C1 20 ED FD 18 69 01 C9 DB DO F6 60
```

(verifica)

```
*300.30C
0300-A 9 C1 20 ED FD 18 69 01
0308-C9 DB DO F6 60
```

(salvataggio)

```
*300.30CW
```

(esecuzione)

```
*300G
ABCDEFGHIJKLMNOPQRSTUVWXYZ
*
```

voi inserite in memoria il programma da eseguire, lo verificate, lo salvate e lo eseguite.
Questo è il procedimento che in generale deve essere eseguito.

Disassemblaggio di un programma in linguaggio macchina. Il Monitor offre all'utente la possibilità di avere un listato del sottoprogramma in linguaggio macchina presente in memoria. Questo listato viene presentato nella forma di istruzioni in linguaggio assemblatore. Invece di dover ricercare le istruzioni secondo i codici che comprendono uno, due o tre bytes, il comando List esegue questa scomposizione e ricerca l'istruzione mnemonica corrispondente e questo per venti istruzioni di fila.
Il formato del comando è il seguente:

{indirizzo d'inizio} L

Il Monitor utilizza il puntatore di programma (registro interno del microprocessore) come puntatore alla prossima istruzione da disassemblare. È necessario quindi precisare l'indirizzo d'inizio soltanto la prima volta allorquando si deve eseguire il disassemblaggio di una zona di memoria.
Nell'esempio che segue, se digitate il comando *300L, il Monitor segnalerà:

```
0300 - A9 C1      LDA #$C1
0302 - 20 ED FD   JSR #FDED
0305 - 18 E       CLC
0306 - 69 01      ADC #$01
0308 - C9 DB      CMP #$DB
030A - DO F6      BNE $0302
030C - 60         RTS
030D - A3         ??  fine del programma, il seguito non ci inte-
                      ressa
030E - 00         BRK
```

Il miniassemblatore. Se siete in possesso di un Apple II standard o di una language card (utilizzare INT), potrete scrivere dei programmi in assemblatore. Esso risiede nella stessa ROM dell'Integer Basic e non è disponibile con l'Applesoft.
Il miniassemblatore vi consente di inserire in macchina i codici mnemonici dell'istruzione. Esso però non accetta i simboli e quindi dovrete fornire gli indirizzi direttamente in forma esadecimale.

* ACCESSO AL MINIASSEMBLATORE

L'entry point del miniassemblatore si trova all'indirizzo $F666. Per accedere al miniassemblatore partendo dal Monitor dovete digitare dunque questo comando:

```
*F666G
```

Per accedere al miniassemblatore partendo da un Basic, sia esso Integer Basic su ROM oppure Applesoft su dischetto o cassetta, dovete richiamare il sottoprogramma che si trova all'indirizzo -2458 (versione decimale dell'indirizzo $666) e dunque digitare:

```
>
]CALL-2458
-
```

Il miniassemblatore visualizzerà allora un punto esclamativo "!".
*Il miniassemblatore esiste ora su dischetto per gli utenti dell'*Apple II + .
È possibile ottenerlo partendo dal file INT *Basic fornito con il* DOS 3.3, *copia delle* ROM *di Integer Basic.*

* ACCESSO AI COMANDI DEL MONITOR PARTENDO DAL MINIASSEMBLATORE

È possibile accedere ai comandi del Monitor partendo dal miniassemblatore. Per ottenere ciò è necessario digitare il carattere $ prima di digitare il vostro comando.
Se volete ad esempio disassemblare le istruzioni che si trovano all'indirizzo $300, vi basterà digitare il comando:

```
!$300L
```

* USCITA DAL MINIASSEMBLATORE

Per uscire dal miniassemblatore utilizzare i comandi del Monitor.
Per ritornare al Basic digitare a seconda dei casi uno dei seguenti quattro comandi:

```
!$CTRL-B
!$CTRL-C
!&3DOG      APPLESOFT caricato partendo da un dischetto
!$OG        APPLESOFT caricato partendo da una cassetta
```

Per ritornare al Monitor utilizzare il comando !$FF69G oppure premete il tasto RESET.

* ISTRUZIONI

Il miniassemblatore gestisce un puntatore alla prossima istruzione da disassemblare che è diverso da quello del Monitor. Dovrete quindi inizializzarlo per introdurre la prima istruzione da disassemblare.
Ad esempio se volete inserire l'istruzione BRK all'indirizzo $300, dovrete utilizzare il seguente comando:

!300:BRK

Se volete far seguire quest'istruzione da un'altra, è inutile calcolare l'indirizzo della nuova istruzione, poiché il miniassemblatore esegue ciò automaticamente. Dovete comunque lasciare uno spazio tra il punto esclamativo e l'istruzione da disassemblare. Il formato delle istruzioni accettate dal miniassemblatore è il seguente:

!codice mnemonico su tre lettere operanti.

Quando la riga è digitata, il miniassemblatore la analizza. Se esso non riscontra alcun errore, incrementa il puntatore alla prossima istruzione e passa la mano all'utilizzatore visualizzando un punto esclamativo. Altrimenti il miniassemblatore emette un "beep" e visualizza un accento circonflesso sotto il primo carattere incriminato. Poi rimane in attesa che l'utente ridigiti l'istruzione.
Il formato dell'operando o degli operandi è imposto a seconda dell'istruzione e del tipo di indirizzamento. Il microprocessore 6502 consente undici tipi di indirizzamento.
Esistono tuttavia sei formati accettati dal miniassemblatore per l'operando o gli operandi.
Essi sono i seguenti:

Tipo d'indirizzamento	Formato	Numero di formato
Accumulatore		
Implicito		
Immediato	#$ valore	1
Assoluto	$ valore	2
Assoluto a indice	$ valore,X	3
	$ valore,Y	4
Pagina zero	$ valore	2
Pagina zero a indice	$ valore,X	3
	$ valore,Y	4
Relativo	$ valore	2
A indice indiretto	($ valore,X)	5
Indiretto a indice	($ valore),Y	6

La scelta tra il tipo assoluto e quello in pagina zero si effettua a seconda del valore dell'operando. Se tale valore è piccolo, viene scelto il tipo di indirizzamento in pagina zero, altrimenti viene scelto il tipo assoluto.

Messa a punto dei programmi. In parallelo al miniassemblatore, la versione standard del Monitor fornisce all'utente due comandi per permettergli di fare eseguire i suoi programmi passo passo, conservando la traccia dei registri.

- Il comando Step permette di eseguire un programma conservando la traccia dei registri.
- Il comando Trace permette di eseguire un programma conservando la traccia dei registri.

Questi due comandi non sono offerti con la ROM autostart.

* ESECUZIONE DI UN PROGRAMMA PASSO PASSO

Per verificare un programma assembler, risulta spesso utile eseguire una parte dello stesso istruzione per istruzione, verificando il risultato dopo l'esecuzione di ogni istruzione. Ciò viene svolto dal comando Step che ha il seguente formato:

{indirizzo dell'istruzione da eseguire} S

Per ogni comando S, il Monitor visualizza l'istruzione da eseguire, sia nel suo formato in linguaggio macchina che in quello mnemonico. Esso la esegue successivamente incrementando il puntatore di programma all'istruzione successiva. Dovete quindi digitare l'indirizzo dell'istruzione solamente per la prima istruzione da eseguire nella serie. Il concatenamento viene eseguito automaticamente.
Quando un'istruzione è stata trattata dal 6502, il Monitor visualizza i registri del microprocessore e passa la mano all'utente. Potete allora modificare i registri. Essi verranno quindi considerati quando verrà digitato il successivo comando S oppure GO oppure Trace.
Se ad esempio volete verificare il seguente programma:

```
300 - A2 02       LDX  #102
302 - B5 00       LDA  00,X
304 - 95 10       STA  10,X
306 - CA          DEX
307 - 8D 30 C0    STA  C030
30A - 10 F6       BPL  0302
30C - 60          RTS
```

dovete digitare il seguente comando:

*300S

Il Monitor visualizzerà:

0300-A2 02 LDX #$02

A=0A X=02 Y=D8 P=30 S=F8

Se desiderate continuare l'esecuzione delle istruzioni successive, dovete digitare il comando S tante volte quante sono le istruzioni da trattare.
È possibile tra due comandi S utilizzare altri comandi del Monitor per esaminare e modificare la memoria o i registri, attivare una stampante, passare alla funzione inversa o normale dello schermo.

* TRACCIA DURANTE L'ESECUZIONE DI UN PROGRAMMA

Per i programmi che sono troppo lunghi da trattare passo passo, il Monitor offre all'utente la possibilità di avere una traccia dell'esecuzione di un programma. Il formato del comando Trace è il seguente:

{indirizzo dell'inizio del programma da 'tracciare'} T

L'indirizzo di inizio non deve essere fornito al Monitor se esso non è stato già posizionato dal comando Step ad esempio. La Trace del programma si arresterà solamente se il Monitor incontra un'istruzione BRK (codice 00) o se avete premuto il tasto Reset.
Il formato della segnalazione è lo stesso che per il comando Step: l'istruzione in codice macchina con il codice mnemonico e gli operandi corrispondenti e i registri alla fine dell'istruzione.
I comandi del Monitor S, L, T, G *utilizzano tutti lo stesso puntatore.*

Esempio di creazione di un programma assemblatore tramite Monitor. Vi forniamo qui di seguito un esempio di messa a punto di un sottoprogramma assemblatore per la visualizzazione di un punto (istruzione PLOT in Basic). Le coordinate vengono fornite dalla lettura delle manopole per giochi, grazie al miniassemblatore e ai comandi del Monitor nella sua versione standard.
Entriamo innanzitutto nel miniassemblatore.
Per fare ciò, digitate CALL-2458 se lavorate in Integer Basic, INT e CALL-2458 se avete la language card. Quando viene visualizzato il punto esclamativo, il miniassemblatore rimane in attesa dei vostri comandi. Digitate allora le seguenti istruzioni:

```
0304-: LDA   $C050          (Return)
0307-: LDA   $C053
030A-: LDA   $C054
030D-: LDA   $C056
0310-: JSR   $FB46
0313-: LDA   #$99
0315-: STA   $30
```

```
0317- : LDX   #$00
0319- ; JSR   $FB1E
031C : CPY    #$27
031E : BCC    $0322
0320 : LDY    #$27
0322 : STY    $0300
0325 : LDA    #$50
0327 : JSR    $FCA8
032A ; LDX    #$01
032C : JSR    $FB1E
032F : CPY    #$27
0331-  BCC    $0335
0333-  .LDY   #$27
0335- : STY   $0301
0338- : LDA   #$50
033A-   JSR   $FCA8
033D-   LDA   $C061
0340-   CMP   #$00
0342- : BPL   $0317
0344- : LDA   $0300
0347-   LDY   $0301
034A : JSR    $F800
034D- : JMP   $0317
0350-   BRK
```

Ritornate al Basic ($CTRL–C) e salvate il vostro programma:

BSAVE BPLOT, A$300, L$50

Passiamo adesso nel Monitor e digitate:

*304L

Il listato ottenuto è il seguente:

```
0304-   AD 50 C0    LDA   $C050
0307-   AD 53 C0    LDA   $C053
030A-   AD 54 C0    LDA   $C054
030D-   AD 56 C0    LDA   $C056
0310-   20 46 FB    JSR   $FB46
0313-   A9 99       LDA   #$99
0315-   85 30       STA   $30
0317-   A2 00       LDX   #$00
0319-   20 1E FB    JSR   $FB1E
031C-   C0 27       CPY   #$27
031E-   90 02       BCC   $0322
0320-   A0 27       LDY   #$27
0322-   8C 00 03    STY   $0300
0325-   A9 50       LDA   #$50
0327-   20 A8 FC    JSR   $FCA8
```

```
032A-   A2 01       LDX   #$01
032C-   20 1E FB    JSR   $FB1E
032F-   C0 27       CPY   #$27
0331-   90 02       BCC   $0335
0333-   A0 27       LDY   #$27
0335-   8C 01 03    STY   $0301
0338-   A9 50       LDA   #$50
033A-   20 A8 FC    JSR   $FCA8
033D-   AD 61 C0    LDA   $C061
0340-   C9 00       CMP   #$00
0342-   10 D3       BPL   $0317
0344-   AD 00 03    LDA   $0300
0347-   AC 01 03    LDY   $0301
034A-   20 00 F8    JSR   $F800
034D-   4C 17 03    JMP   $0317
0350-   00          BRK
```

Potete adesso eseguire il vostro programma. Per fare ciò digitate il comando:

*304G

Lo schermo passa alla funzione grafica a bassa risoluzione e il punto di coordinate PDL (0), PDL (1) viene visualizzato quando premete il pulsante della manopola 0.
Siccome l'ultima istruzione esegue un GOTO all'inizio del programma dopo le inizializzazioni, l'unico modo di sospenderne l'esecuzione è di premere il tasto Reset.
A titolo di paragone, ecco il listato del programma con i commenti ottenuti sulla versione 1.5 dell'assemblatore LISA.

```
0300     1          ORG $300
0300     2  ;
0300     3  ; DEFINIZIONE DEI SOTTOPROGRAMMI
0300     4  ;
0300     5  WAIT    EQU $FCA8
0300     6  PDL     EQU $FB1E
0300     7  PLOT    EQU $F800
0300     8  ;
0300     9  ; DEFINIZIONE DEGLI INDIRIZZI DI MEMORIA
0300    10  ;
0300    11  GRAPH   EQU $C050
0300    12  MIXTE   EQU $C053
0300    13  PAGE1   EQU $C054
0300    14  BASRES  EQU $C056
0300    15  INIZIO  EQU $C061
0300    16  ORANGE  EPZ $99
0300    17  ACOLOR  EPZ $30
0300    18  ;
```

```
0300           19  ;  DEFINIZIONE DELLE VARIABILI
0300           20  ;
0301           21  LINEA   DFS $1
0302           22  COLON   DFS $1
0302           23  ;
0302           24  ;  INIZIALIZZAZIONE
0302           25  ;
0304           26          ORG $304
0304 AD50C0    27          LDA GRAPH
0307 AD53C0    28          LDA MIXTE
030A AD54C0    29          LDA PAGE1
030D AD56C0    30          LDA BASRES
0310 2046FB    31          JSR $FB46
0313 A999      32          LDA #ORANGE
0315 8530      33          STA ACOLOR
0317           34  ;
0317           35  ; LETTURA DELLA LINEA
0317           36  ;
0317 A200      37  ANELLO LDX #$00
0319 201EFB    38          JSR PDL
031C C027      39          CPY #$27
031E 9002      40          BCC OK1
0320 A027      41          LDY #$27
0322 8C0003    42  OK1     STY LINEA
0325           43  ;
0325           44  ;  LOOP DI ATTESA
0325           45  ;
0325 A950      46          LDA #$50
0327 20A8FC    47          JSR WAIT
032A           48  ;
032A           49  ;  LETTURA DELLA COLONNA
032A           50  ;
032A A201      51          LDX #$01
032C 201EFB    52          JSR PDL
032F C027      53          CPY #$27
0331 9002      54          BCC OK2
0333 A027      55          LDY #$27
0335 8C0103    56  OK2     STY COLON
0338           57  ;
0338           58  ;  LOOP DI ATTESA
0338           59  ;
0338 A950      60          LDA #$50
033A 20A8FC    61          JSR WAIT
033D           62  ;
033D           63  ;  TEST PULSANTE PREMUTO
033D           64  ;
033D AD61C0    65          LDA INIZIO
0340 C900      66          CMP #$00
0342 10D3      67          BPL ANELLO
0344           68  ;
0344           69  ;  VISUALIZZAZIONE DEL PUNTO
0344           70  ;
0344 AD0003    71          LDA LINEA
0347 AC0103    72          LDY COLON
034A 2000F8    73          JSR PLOT
034D 4C1703    74          JMP ANELLO
```

```
0350 00         75          BRK
                76          END
***** END OF ASSEMBLY


            *************************
            *                       *
            * SYMBOL TABLE -- V 1.5 *
            *                       *
            *************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

ORANGE 0099  ACOLOR 0030

** ABSOLUTE VARABLES/LABELS

WAIT   FCA8  PDL    FB1E  PLOT   F800  GRAPH  C050
MIXTE  C053  PAGE1  C054  BASRES C056  INIZIO C061  LINEA  0300  COLON  0301
ANELLO 0317  OK1    0322  OK2    0335

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:008A
```

• Comandi che consentono di modificare l'ingresso-uscita

Funzione dello schermo. I comandi I/N consentono di far passare lo schermo dalla funzione inversa a quella normale. Tuttavia, i caratteri prelevati dalla tastiera vengono visualizzati con la funzione normale.

Utilizzo di una stampante. La stampante è utilizzata digitando il comando CTRL-P. La visualizzazione sullo schermo viene quindi bloccata in quanto i caratteri da visualizzare vengono inviati alla stampante.
Il formato esatto del comando è il seguente:

numero del connettore CTRL-P

in cui il numero del connettore è compreso tra 1 e 7 e in cui il connettore indicato contiene una scheda di interfaccia per stampante.
Se il connettore che indicate non contiene una scheda di espansione, perderete il controllo del vostro Apple.
Per ritornare al modo di funzionamento normale dell'Apple dovrete digitare il comando CTRL-P.

Utilizzo di un'altra tastiera diversa da quella dell'Apple (o di qualunque altra periferica che consente l'input dei dati). È possibile utilizzare un'altra sorgente di dati per sostituire la tastiera dell'Apple. Per fare ciò è sufficiente avere una scheda di interfaccia e digitare il comando:

numero del connettore della scheda CTRL-K

Per ritornare al modo di funzionamento normale dell'Apple, dovrete digitare il comando 0 CTRL-K.
Se digitate un numero di connettore errato, perderete il controllo del vostro Apple.

- **Creazione del vostro comando utente**

Il comando CTRL-Y innesca il salto del Monitor all'indirizzo $3F8. A quest'indirizzo potete inserire un'istruzione Jump al nostro programma.
Potete ad esempio decidere di utilizzare questa possibilità per passare nel miniassemblatore. Vi sarà sufficiente, a questo scopo, digitare il seguente comando:

3F8:4C 66 F6 (cioè JMP $F666)

Quando digiterete CTRL-Y, vi ritroverete nel miniassemblatore.
Vi è anche consentito di sfruttare questa possibilità per ritornare al Basic caricato in memoria mediante un dischetto. A questo scopo è sufficiente digitare:

3F8:4C D0 03 (cioè JMP 3D0 oppure 3DOG).

4.3.4 Indirizzi di memoria e sottoprogrammi del Monitor

Questo paragrafo descrive: la scheda di memoria dell'Apple II, i sottoprogrammi d'ingresso-uscita con la tastiera e l'uscita video dell'Apple, i sottoprogrammi di gestione degli interrupts, l'utilizzo dei programmi del Monitor come sottoprogrammi ed infine la gestione delle altre operazioni di ingresso-uscita dell'Apple II.

- **Descrizione della scheda di memoria**

Presentazione generale. La memoria è suddivisa in segmenti di 256 bytes, chiamati pagine, numerati partendo dallo 0. La pagina 0 è riservata al Monitor, al DOS e al Basic; la pagina 1 è utilizzata come pila dal microprocessore 6502 (noi la utilizzeremo soltanto in questo modo). La pagina 2 serve per bufferizzare i dati prelevati dalla tastiera. Vengono di seguito le pagine da 4 a 7 utilizzate per la pagina grafica numero 1 a bassa risoluzione, le pagine da 8 a B utilizzate con la seconda pagina grafica a bassa risoluzione. Le due pagine grafiche ad alta risoluzione si trovano rispettivamente agli indirizzi $2000-$3FFF e $4000-$7FFF. La memoria ROM contenente il Monitor si trova tra gli indirizzi $D000-$FFFF. La tabella che segue ricapitola i diversi indirizzi.

Scheda di memoria dell'Apple II

Tabella locazioni di memoria dell'Apple II

0000-00FF	Pagina 0		
0100-01FF	Pila (Stack) del 6502		
0200-02FF	Buffer ingresso tastiera		
0300-03CF 03D0-03EF 03F0-03FF	Vettore dei puntatori DOS		
0400-07FF	Pagina 2 per grafica a bassa risoluzione		
0800-0BFF	Programmi utente e puntatori	Pagina 2 per grafica in bassa risoluzione	RAM utilizzata da Applesoft caricato da disco
0C00-2000	Programma utente Basic Applesoft in ROM (Apple II+)		
2000-2FFF 3000-3FFF	Puntatori Basic Limite 16 K di memoria	Pagina 1 grafica Alta risoluzione	Programma utente RAM Applesoft
4000-5FFF		Pagina 2 grafica Alta risoluzione	
-7FFF	Limite 32 K	Memoria	
-BFFF	Limite 48 K		
C000-CFFF	Zona schede I/O		
D000-FFFF	ROMS(s) BASIC o MONITOR		

Utilizzo della pagina zero ($0000, $00FF). Il Monitor utilizza nella pagina zero gli indirizzi compresi tra $20 e $49 per eseguire le funzioni generali, tra $4E e $4F per leggere da tastiera e, per la versione Integer Basic (vecchio Monitor), gli indirizzi da $50 a $55.
Descriviamo qui di seguito e più precisamente l'utilizzo di questi indirizzi dal Monitor dell'Apple II+.

Indirizzo decimale	Indirizzo esadecimale	Etichetta	Utilizzo
00-01	$00-$01	LOC0 LOC1	Questi indirizzi sono utilizzati dalla ROM autostart all'accensione della macchina per determinare la presenza o no di un flag e per caricare il DOS.
32	$20	WNDLFT	Colonna d'ingresso (a sinistra sullo schermo). Viene utilizzata dal sottoprogramma VTABZ che posiziona BASL,H da WNDLFT e CV. Dopo la modifica del contenuto dell'indirizzo, il risultato sarà operativo al prossimo utilizzo.
33	$21	WNDWDTH	Larghezza della finestra sullo schermo. È utilizzata dal sottoprogramma COUT per determinare il passaggio alla linea seguente.
34	$22	WNDTOP	Prima linea dello schermo. 0-22 per utilizzo in modo testo. 20-22 per utilizzo in modo testo-grafica in bassa risoluzione. Si usa per indicare la linea da cancellare o da spostare quando si aggiunge una linea in basso sullo schermo. Segnala anche il numero di linea dopo un HOME.
35	$23	WNDBTM	È l'ultima linea dello schermo. In effetti contiene il numero di riga seguente. Finestra <-WNDBTM-> Il suo valore è compreso fra (WNDTOP) + 1 e 24. È testato dopo un CR (Ritorno carrello) e un LF (Linea seguente).
36	$24	CH	Tabulazione orizzontale del cursore all'apertura della finestra. Indica la posizione del prossimo carattere da inserire. Il suo valore deve essere compreso fra 0 e (WNDWDTH) —1. Si passerà automaticamente alla linea successiva superando il limite superiore. Questa locazione è utilizzata dai sotto programmi d'inserimento dei caratteri sullo schermo.

(segue)

(seguito)

Indirizzo decimale	Indirizzo esadecimale	Etichetta	Utilizzo
37	$25	CV	Posizione del cursore in rapporto al punto più alto dello schermo (e non della finestra). Questa locazione è utilizzata solo per determinare BASL e H. Se il valore di questo indirizzo indica la linea inferiore al limite WNDBTM della finestra tutte le linee di questa saranno traslate verso l'alto di una linea finché la linea corrente sarà completata.
38-39	$26-$27	GBASL GBASH	Indirizzo di memoria dove sarà inserito il prossimo punto in bassa risoluzione.
40-41	$28-$29	BASL BASH	Indirizzo di memoria dove sarà inserito il prossimo carattere. È funzione di CV e WNDLFT. Ed è posizionata da BASCALC.
42-43	$2A-$2B	BAS2L BAS2H	Zona di lavoro dopo lo spostamento verso l'alto delle linee della finestra.
44-45	$2C-$2D	 H2 V2 LMNEM RMNEM RTNL RTNH	Molteplici utilizzi. Ultimo punto inseribile sulla linea indicata da HLINE (0-39). Puntatore utilizzato dal disassemblatore nella tavola dei mnemonici 6502. Utilizzando dal comando TRACE per conservare il modo dello schermo. 0-39 testo e bassa risoluzione. 0-47 alta risoluzione. Memorizza l'ultima linea segnalata da VLINE (Tabulatore verticale).
46	$2E	 FORMAT CHKSUM MASK	Molteplici utilizzi. Utilizzato dal disassemblatore per memorizzare un codice specifico dell'istruzione in corso. Utilizzato per calcolare il codice di controllo nella lettura da cassetta. Questa locazione è utilizzata da PLOT per verificare se il punto è sulla linea superiore o inferiore dell'indirizzo puntato da GBASL,H. MASK = $0F MASK = $F0

(segue)

(seguito)

Indirizzo decimale	Indirizzo esadecimale	Etichetta	Utilizzo
47	$2F		Molteplici utilizzi.
		LENGHT	Lunghezza dell'istruzione utilizzata dal disassemblatore.
		SIGN	Il bit 1 è posto a 1 se il risultato di a MULPM o DIVPM (moltiplicazione o divisione 16 bit, vecchio monitor) è complemento del programma chiamante.
		LASTIN	Utilizzato da RDBIT come zona lavoro quando viene utilizzata la cassetta.
48	$30	COLOR	Codice colore utilizzato da SET COL, COLOR. Il suo valore è 17 * Codice indicato nel Basic.
49	$31	MODE	
50	$32	INVFLG	Utilizzo dello schermo: $FF modo normale (nero su bianco). $3F modo inverso (bianco su nero). $7F modo lampeggiante. Utilizzato da COUT1 e può essere settato da SETNORM, RESET, SETINV. Non esiste sottoprogramma che setti il modo lampeggiante.
51	$33	PROMPT	Questo byte contiene il carattere utilizzato da GETLN per i dati da tastiera.
52 53	$34 $35	YSAV YSAV1	Zona salvataggio del registro Y da parte del monitor e di COUT.
54 55	$36 $37	CSWL CSXH	Indirizzo del sottoprogramma di visualizzazione dei caratteri sullo schermo. È posizionato per défaut all'indirizzo COUT1. Viene utilizzato se avete un altro schermo-tastiera o per visualizzare dati. Il comando n CTRL-P fornisce all'indirizzo il valore Cnoo (Scheda espansione n.).
56 57	$38 $39	KSWL KSWH	Lo stesso nella scelta dei dati. Il comando di monitor utilizzato è n CTRL-K Per défaut il valore di questo indirizzo è quello del sottoprogramma KEYIN.

(segue)

(seguito)

Indirizzo decimale	Indirizzo esadecimale	Etichetta	Utilizzo
58 59	$3A $3B	PCL PCH	Zona utilizzata per salvare il puntatore alla prossima istruzione da eseguire (PC). È utilizzato dal miniassembler per riconoscere l'istruzione seguente. I comandi L, G, S, T usano o posizionano questo indirizzo. Questi 2 byte contengono anche a seguito di un BRK l'indirizzo dell'istruzione che segue BRK.
60-67	$3C $3D-$43	XQT XQTNZ	Zona di 8 byte di lavoro per il monitor. Si usa normalmente per i seguenti comandi: S } Vecchio T } Monitor SUBTRACT MOVE VERIFY READ WRITE STORE Esame memoria.
69 70 71 72 73	$45 $46 $47 $48 $49	ACC XREG YREG STATUS SPNT	Salvataggio dell'accumulatore. Salvataggio del registro X. Salvataggio del registro Y. Salvataggio del registro di stato (flag). Salvataggio del puntatore di pila. Il sottoprogramma SAVE realizza queste operazioni. Il sottoprogramma RESTORE li ripristina.
74-77	$4A-$4D		Zona non utilizzata.
78-79	$4E-$4D	RNDL-RNDH	Contatore su 16 bit incrementato da KEYIN per testare i tasti di tastiera.
80 81 82 83 84 85	$50 $51 $52 $53 $54 $55	ACL ACH XTNDL XTNDH AUXL AUXH	Zona utilizzata unicamente dai sottoprogrammi e divisione del vecchio monitor.

Pagine da 1 a 3.

* Pagina 1 ($0100-$01FF)

La pagina 1 è la zona della pila 6502. Il Monitor utilizza questa zona solamente attraverso le istruzioni 6502 che utilizzano la pila (PHA, JSR, RIS...). Il Monitor non inizializza mai né la zona né il puntatore della pila (registro S).

* Pagina 2 ($0200-$02FF)

La pagina 2 è il buffer dove sono allocate le informazioni prelevate da tastiera mediante il sottoprogramma GETLN. Questo inizializza il registro X come registro indice, poi alloca all'etichetta ADDINP il carattere all'indirizzo $0200+X. Al ritorno da GETLN, l'ultimo carattere nel buffer è 'RETURN' (codice $8D = $80 + $0D).

* Pagina 3 ($0300-$03FF)

La pagina 3 contiene dei vettori di indirizzi utilizzati dagli interrupts. Questi vettori sono allocati nella parte superiore della pagina 3.
Gli indirizzi da $03D0 e $03EF sono utilizzati dal DOS. Il resto della pagina è utilizzabile dai programmi assemblatori.
La tabella che presentiamo descrive i vettori di interrupt.

Indirizzo esadecimale	Indirizzo decimale	Utilizzo
$0300-$03CF	768-995	Zona libera
$03D0-$03EF	995-1007	DOS (cfr. volume 2, § 1)
$03F0-$03F1	1008-1009	Vettore d'interruzione BRK (indirizzo)
$03F2-$03F3	1010-1011	Vettore d'interruzione RESET usato per l'autostart (indirizzo)
$03F4	1012	Indicatore di caricamento del sistema: - Se ($A5) o (($03F3)) = ($03F4) contenuto di 3F3 Si ricarica completamente il sistema, compreso il DOS. - Altrimenti si ferma il programma in corso:
$03F5-$03F7	1013-1015	Riservato dall'APPLESOFT (istruzione)
$03F8-$03FA	1016-1018	Quando l'utente digita CIRL-Y
$03FB-$03FD	1019-1021	Vettore delle interruzioni non mascherabili (istruzione da eseguire) (NMI)
$03F4-$03FF	1022-1023	Vettore d'interruzione IRQ (indirizzo di salto)

Pagine di testo e grafica a bassa risoluzione. Gli indirizzi compresi tra $0400 e $07FF costituiscono la prima pagina di testo e di grafica a bassa risoluzione. È a questi indirizzi che vengono ricopiate le informazioni da

visualizzare sullo schermo. Queste ultime saranno visualizzate mediante una logica hardware per la gestione dello schermo.
All'indirizzo $0800 inizia generalmente il programma utente.
Tuttavia, per i sistemi forniti di RAM, la zona compresa tra gli indirizzi $0800 e $0BFF può essere utilizzata come seconda pagina di testo e di grafica a bassa risoluzione. Tale pagina è accettata dalla logica hardware per la gestione dello schermo, ma non dal Monitor.
Per passare dalla pagina 1 alla pagine 2, è sufficiente digitare:

POKE-16299,0

Per eseguire l'operazione inversa è sufficiente digitare:

POKE-16300,0

La tabella che riportiamo di seguito, evidenzia gli indirizzi in cui sono situate le righe dello schermo:

Linea	Pagina 1		Pagina 2	
	Decimale	**Esadecimale**	**Decimale**	**Esadecimale**
0	1024	$0400	2048	$0800
1	1152	$0480	2176	$0880
2	1280	$0500	2304	$0900
3	1408	$0580	2432	$0980
4	1536	$0600	2560	$0A00
5	1664	$0680	2688	$0A80
6	1792	$0700	2816	$0B00
7	1920	$0780	2944	$0B80
8	1064	$0428	2088	$0828
9	1192	$04A8	2216	$08A8
10	1320	$0528	2344	$0928
11	1448	$05A8	2472	$09A8
12	1576	$0628	2600	$0A28
13	1704	$06A8	2728	$0AA8
14	1832	$0728	2856	$0B28
15	1960	$07A8	2984	$0BA8
16	1104	$0450	2128	$0850
17	1232	$04D0	2256	$08D0
18	1360	$0550	2384	$0950
19	1488	$05D0	2512	$09D0
20	1616	$0650	2640	$0A50
21	1744	$06D0	2768	$0AD0
22	1872	$0750	2896	$0B50
23	2000	$07D0	3024	$0BD0

I 40 caratteri di una riga sono raggruppati in memoria. La riga numero L della pagina P si trova all'indirizzo:

$$ADDRL = 1024*P + 256((L/2)MOD4) + (128*(LMO2)) + 40((L/8)MOD4)$$

Possiamo osservare che 24 righe di 40 caratteri occupano 960 bytes. Tuttavia 1024 bytes sono riservati in memoria per semplificare l'organizzazione hardware. 64 bytes sono dunque disponibili in ogni pagina a gruppi di 8 bytes.
Questi bytes non rimangono disponibili nella pagina 1. Essi sono utilizzati dalle schede di espansione nel seguente modo:

Comune	**slot 1**	**slot 2**	**slot 3**	**slot 4**	**slot 5**	**slot 6**	**slot 7**
$0478	$0479	$047A	$047B	$047C	$047D	$047E	$047F
$04F8	$04F9	$04FA	$04FB	$04FC	$04FD	$04FE	$04FF
$0578	$0579	$057A	$057B	$057C	$057D	$057E	$075F
$05F8	$05F9	$05FA	$05FB	$05FC	$05FD	$05FE	$05FF
$0678	$0679	$067A	$067B	$067C	$067D	$067E	$067F
$06F8	$06F9	$06FA	$06FB	$06FC	$06FD	$06FE	$06FF
$0778	$0779	$077A	$077B	$077C	$077D	$077E	$077F
$07F8	$07F9	$07FA	$07FB	$07FC	$07FD	$07FE	$07FF

L'indirizzo 2040 ($07F8) *è riservato e deve contenere* Cn *in cui* n *è l'indirizzo della periferica attiva. Questo consente di gestire le interruzioni.*
Vi sconsigliamo di salvare in blocco gli indirizzi da $0400 *a* $07FF *perché quando dovrete riutilizzare questa zona, gli indirizzi riservati alle periferiche saranno distrutti e ciò potrebbe essere fonte di malfunzionamenti.*

Memoria riservata per l'input-output. A parte i 64 bytes definiti in precedenza, una grande zona di memoria è riservata per le operazioni di input-output, compresa tra gli indirizzi $C000 e $CFFF. La descriveremo in questo paragrafo.

* INDIRIZZI SPECIALI

Verranno esaminati in questa sede i bytes utilizzati dalla tastiera, dalla cassetta, dall'altoparlante, dalle manopole di gioco e dalla scelta della grafica.

TASTIERA:

la tastiera utilizza due bytes situati tra gli indirizzi $C000 e $C010 che hanno le seguenti funzioni:

$C000 (-16384) 1 bit di stato (carattere ricevuto)
7 bits di dati (codice ASCII standard).

*Questo fa sì che i codici trattati dall'*Apple II *siano compresi tra* 128 *e* 255 *se il bit di stato non sia stato risettato a* 0.

$C010 (−16368) scrivere a quest'indirizzo significa che il carattere è stato letto. Il bit di stato viene allora resettato a 0.

CASSETTA:

la gestione della cassetta utilizza due bytes in memoria che si trovano agli indirizzi:

$C060 (−16288):IN ingresso
$C020 (−16352):OUT uscita

Descriviamo quindi il funzionamento della cassetta.
Per registrare delle informazioni binarie (0,1) su una cassetta, viene utilizzato il byte di indirizzo $C020. A quest'indirizzo si trova un flag (logico) che conserva uno stato e che cambia stato ogni volta che ad esso si fa riferimento in un'istruzione.
Una lettura del valore contenuto a quest'indirizzo, provoca un cambiamento di stato, una registrazione ne provoca due. Invero, una registrazione in memoria eseguita dal 6502 è sempre preceduta da una lettura. Esiste tra il flip-flop e la presa OUT una logica hardware che trasforma i cambiamenti di stato logici (0,1), in variazioni della tensione di uscita (0/V−25m V). In questo modo le informazioni sono immagazzinate sulla cassetta. Come fa l'Apple II a rileggerle? Esiste a questo scopo una logica hardware che permette la decodifica delle variazioni di tensione in ingresso onde generare dei valori binari 0 e 1. Un programma può testare il valore generato, leggendo l'indirizzo $C060. Se il valore ottenuto è superiore a 128, lo stato è 1, altrimenti è 0.
La lettura dei dati deve essere fatta in assembler, perché il Basic è fuori gioco in quanto a rapidità.

ALTOPARLANTE:

ogni riferimento all'indirizzo $C030 (−16336), provocherà l'emissione di un "beep". Allo stesso modo che per la cassetta, una registrazione cambierà per due volte lo stato del flip-flop a quest'indirizzo e quindi lo stato del flip-flop sarà quello preesistente alla registrazione.

MANOPOLE DI GIOCO:

il connettore delle manopole di gioco fornisce:

1. tre ingressi numerici su un bit 0/1;
2. tre uscite numeriche su un bit 0/1;
3. quattro ingressi analogici;
4. un segnale di validazione dei dati.

Potete utilizzare questo connettore per altri tipi di utilizzo (relais, altoparlanti...) e controllarlo mediante programma.

1. Ingressi numerici: vengono accettati 3 ingressi digitali. I loro stati possono essere letti (allo stesso modo che per una cassetta) agli indirizzi $C061, $C062, $C063 (per esempio da -16287 a -16285). Se essi sono negativi, il pulsante è premuto.
2. Uscite numeriche: queste sono gestite da flags logici che utilizzano due bytes di memoria. La lettura del primo byte pone il flag in un determinato stato, quella del secondo byte lo porrà in quello opposto.
Gli indirizzi utilizzati sono i seguenti:

0	OFF	$C058	(-16296)
	ON	$C059	(-16295)
1	OFF	$C05A	(-16294)
	ON	$C05B	(-16293)
2	OFF	$C05C	(-16292)
	ON	$C05D	(-16291)
3	OFF	$C05E	(-16290)
	ON	$C05F	(-16289)

OFF corrisponde a { 0 logico / 0 Volt di tensione d'uscita }
ON corrisponde a { 1 logico / 5 Volt di tensione d'uscita }

3. Inputs analogici. I 4 inputs analogici possono essere connessi a dei potenziometri di 150 KOhm. Per leggere gli inputs analogici, un programma assemblatore deve seguire questo procedimento:

- Risettare a 0 (RESET) i circuiti contatori mediante la lettura del contenuto dell'indirizzo $C070 (-16272). I valori contenuti agli indirizzi da $C064 a $C067 (da -16284 a -16281), sono quindi superiori a 128.
- Osservare il tempo necessario affinché questi valori si riportino al di sotto di 128. La durata è proporzionale al valore prelevato dal potenziometro e posto tra 5 Volt e la tensione d'ingresso. Il tempo massimo è di 3,060 ms.

Se non è presente un potenziometro all'ingresso o se il valore della resistenza è troppo elevato, i valori letti non raggiungeranno mai lo 0.

4. Segnale di validazione Strobe in uscita: questo segnale consente di validare i dati in uscita dall'Apple II. Esso è controllato facendo riferimento all'indirizzo $C040. Questo provoca il passaggio del segnale da 5 Volt a 0 Volt nello spazio di mezzo secondo.

GRAFICHE:

Tre tipi di informazione possono essere visualizzati sullo schermo:

- testo di 24 righe di 40 caratteri ciascuna;
- grafica a bassa risoluzione:
48 righe di 40 quadrati ciascuna di diversi colori;

- grafica ad alta risoluzione:
192 (altezza) × 28 (larghezza) punti sullo schermo.

Come passare da un tipo di informazione ad un altro?
Esistono 4 flags logici a due valori che consentono la selezione del tipo di informazione. Essi si trovano in corrispondenza ai seguenti indirizzi:

Indirizzo esadecimale	**Indirizzo decimale**	**Descrizione**
$C050	—16304	Tipo grafico
$C051	—16303	Tipo testo
$C052	—16302	Testo o grafica non misti
$C053	—16301	Testo e grafica misti
$C054	—16300	Pagina 1
$C055	—16299	Pagina 2
$C056	—16298	Bassa risoluzione
$C057	—16297	Alta risoluzione

Esistono 10 combinazioni di questi flags logici:

Pagina 1			Pagina 2		
Schermo	**Indirizzi**		**Schermo**	**Indirizzi**	
Testo	$C054		Testo	$C055	
	$C051			$C051	
Bassa risoluzione	$C054	$C056	Bassa risoluzione	$C055	$C056
	$C052	$C050		$C052	$C050
Alta risoluzione	$C054	$C057	Alta risoluzione	$C055	$C057
	$C052	$C050		$C052	$C050
Testo e bassa risolu-	$C054	$C056	Testo e bassa riso-	$C055	$C056
zione	$C053	$C050	luzione	$C053	$C050
Testo e alta risoluzio-	$C054	$C057	Testo e alta risolu-	$C055	$C057
ne	$C053	$C050	zione	$C053	$C050

Digitate il seguente programma:

```
10 A = PEEK (-16299)
20 A = PEEK (-16297)
30 A = PEEK (-16301)
40 A = PEEK (-16304)
```

e quindi eseguitelo.

Verrà visualizzata sullo schermo sia la parte di testo sia la parte grafica ad alta risoluzione.
Digitate di seguito:

A = PEEK (-16302)

Le righe del testo saranno allora soppresse.
Digitate ancora:

A = PEEK (-16303)

La grafica ad alta risoluzione lascerà lo schermo al testo.
Digitate infine:

A = PEEK (-16300)

Ripasserete sulla pagina 1 e vedrete quello che avete digitato precedentemente.

* SCHEDE DI ESPANSIONE

Oltre agli 8 gruppi di 8 bytes definiti nella pagina 1, le schede di espansione dispongono ciascuna di:

- 16 indirizzi riservati per l'input-output (buffer...);
- 256 bytes di ROM contenenti i drivers (controllori di periferiche).

Questi programmi sono preferibilmente indipendenti dal "slot" nel quale viene inserita la scheda. Per determinare quale sia il slot attivo, potrete utilizzare il seguente programma:

```
20  4A  FF   JSR $FF4A
78           SEI
20  58  FF   JSR $FF58
BA           TSX
BD  00  01   LDA $0100,X
8D  F8  07   STA $07F8
29  OF       AND $0F
A8           TAY
```

Il numero del connettore (slot) si trova nel registro Y.
Le tabelle riportate a pagina seguente, riassumono questi indirizzi.
Esiste una zona compresa tra gli indirizzi $C800 e $CFFF riservata ad una ROM (oppure una PROM) di 2 K bytes. Questa ROM può trovarsi su qualsiasi scheda di espansione; essa sarà connessa al bus indirizzi quando la scheda di espansione sarà selezionata.

Ciò consente di avere dei controllori di periferica più completi.
Una scheda di espansione, quando vuole utilizzare questa zona, deve indicare alle altre schede che è in procinto di connettersi. Per ottenere ciò fate riferimento all'indirizzo $CFFF.

	$0	$1	$2	$3	$4	$5	$6	$7	$8	$9	$A	$B	$C	$D	$E	EF
$C080	Ingresso/uscita			0												
$C090	dello slot numero			1												
$C0A0				2												
$C0B0				3												
$C0C0				4												
$C0D0				5												
$C0E0				6												
$C0F0				7												

	$00	$10	$20	$30	$40	$50	$60	$70	$80	$90	$A0	$B0	$C0	$D0	$E0	$F0
$C100	Indirizzo PROM			1												
$C200	dello slot numero			2												
$C300				3												
$C400				4												
$C500				5												
$C600				6												
$C700				7												

• Sottoprogrammi di input-output del Monitor

Esamineremo in sequenza i sottoprogrammi che gestiscono la tastiera, lo schermo e la grafica, la cassetta, le manopole di gioco ed infine l'altoparlante dell'Apple II + .

Gestione della tastiera. I sottoprogrammi di gestione della tastiera realizzano le seguenti operazioni:
- prelevamento di un carattere;
- eco di un carattere;
- memorizzazione di un carattere nel buffer.

La tabella che definiamo nella pagina seguente, descrive i diversi sottoprogrammi (indirizzo, parametri di chiamata, di ritorno, tipo, struttura interna...).

Gestione dello schermo. Vengono offerte dal Monitor diverse funzioni di gestione dello schermo. Esamineremo in cascata i seguenti punti:

- visualizzazione del testo nella finestra definita sullo schermo;
- controllo della funzione di schermo, del tipo di informazione (testo, grafica);
- funzioni di manipolazione dei dati nella finestra (cancellazione, spostamento...);

Sottoprogrammi di gestione della tastiera

Nome	Indirizzo		Parametri di chiamata	Parametri di ritorno	Struttura interna	Semantica
	Decimale	Esa-decimale				
GETLNZ	—665	$FD67	CV = linea da acquisire	A = CR ($8D) X = n° di caratteri acquisiti prima del CR Y = (WNDWDTH) CH = 0 CV = numero locale	Emissione del carriage return (*via* CROUT) Salto all'etichetta GETLN	Passaggio alla linea seguente e acquisizione di una nuova linea.
GETLN	—662	$FD6A	CH = colonna del carattere di intestazione CV = n° linea	idem GETLNZ	GETLN visualizza il carattere di intestazione e inizializza il registro X come indice di memorizzazione nel buffer. Il controllo viene passato a NXTCHAR	Visualizzazione del carattere di intestazione (* monitor,]APPLESOFT, > Integer Basic, ?INPUT, ...) e acquisizione di una linea
NXTCHAR	—651	$FD75	X = $0200 CV, CH	idem GETLNZ	Chiamata di RDCHAR per acquisire il prossimo carattere: - se il carattere è CTRL–V (i.e. →), cerca nella memoria di schermo il carattere che si conserva - se il carattere è una lettera minuscola, trasformazione in maiuscola - se il carattere è un carriage return, salto a CLREOL e, al ritorno, a COUT (sbiancamento del resto della linea e visualizzazione del carriage return) - altrimenti salto a NOTCR	Acquisizione di una linea senza preliminare visualizzazione

(segue)

(seguito)

Nome	Indirizzo		Parametri di chiamata	Parametri di ritorno	Struttura interna	Semantica
	Decimale	Esa-decimale				
NOTCR	—707	$FD3D			Memorizzazione del modo schermo (normale, inverso, ...) Visualizzazione del carattere in modo normale Restauro del modo schermo Testo del carattere: * Backspace (←) salto a BCKSPC * Line cancel (CTRL–X) salto a CANCEL - altrimenti test sul riempimento di una linea. Nel caso in cui vengano acquisiti più di 247 caratteri, chiamata di BELL per prevenire Passaggio a NOTCR1	
NOTCR1	—673	$FD5F			Incremento del registro X e: - chiamata di CANCEL se superamento di linea - altrimenti ritorno a NXTCHAR per il trattamento del carattere seguente	
CANCEL	—670	$FD62			Visualizzazione/*via* COUT Ritorno a GETLNZ per reinizializzare la linea	Sbiancamento linea corrente
BCKSPC	—655	$FD71			Alla chiamata Backspace è già stato visualizzato - se il contenuto del registro X è nullo, ritorno a GETLNZ per reinizializzare la linea; - altrimenti decrementare il registro X	

(segue)

(seguito)

Nome	Indirizzo		Parametri di chiamata	Parametri di ritorno	Struttura interna	Semantica
	Decimale	Esa-decimale				
RDCHAR	—715	$FD35	CH, CV, BASL, H (cfr. pagina 0)	A = carattere acquisito Y = CH X non toccato CV, CH, BASL, H non modificato salvo per le funzioni ESC A B C D	Chiamata di RDKEY per acquisire il carattere: - se il carattere è ESC (ESCAPE): * trasferimento del controllo a ESC1 (vecchio monitor) ESCNEW (Autostart)	Acquisizione di un carattere alla tastiera e realizzazione funzioni ESC (modifica cursore)
RDKEY	—756	$FD0C	CH, CV, BASL, H (cfr. pagina 0)	A = carattere acquisito	Annulla la visualizzazione del carattere situato sotto il cursore, chiama il sottoprogramma che prende fisicamente il carattere e ripristina il carattere sotto il cursore (s.p. all'indirizzo (KSWL, H)).	Acquisizione di un carattere
KEYIN	—741	$FD1B	A = carattere da ripristinare	A = carattere letto	Legge il carattere alla tastiera e ripristina il carattere situato sotto il cursore	

- visualizzazione del testo all'esterno della finestra;
- utilizzo delle seconde pagine grafiche;
- utilizzo della grafica a bassa risoluzione;
- ecc...

* VISUALIZZAZIONE DEL TESTO NELLA FINESTRA

Questo è il tipo di funzionamento standard dell'Apple II.
La visualizzazione di un carattere si fa mediante la chiamata del sottoprogramma COUT, che salta all'indirizzo contenuto in CSWL,H. Quest'indirizzo è posizionato convenzionalmente (per défault) all'atto del Reset insieme all'indirizzo di COUT1 che è il sottoprogramma di visualizzazione di un carattere sullo schermo.
L'utente può cambiare periferica mediante i comandi nCTRL-P (Monitor) oppure PRn (Basic) i quali cambiano CSWL,H # 1.
I seguenti bytes gestiscono la posizione del cursore sullo schermo:

CV	riga del cursore
CH	colonna del cursore colonna di inizio della finestra di sinistra
BASL,H	indirizzo del cursore nella memoria dello schermo
WNDLFT	colonna di inizio della finestra a sinistra
WNDWDTH	larghezza della finestra
WNDTOP	riga superiore della finestra
WNDBTM	riga posta appena al di sotto della finestra

* GESTIONE DEI TIPI DI FUNZIONAMENTO DELLO SCHERMO

Tabella dei sottoprogrammi utilizzati

Funzione	Indirizzo esa-decimale	indirizzo decimale	Nome	Registri distrutti
Salto dell'indirizzo contenuto in CSWL,H	$FDED	—531	COUT	nessuno
Visualizzazione di un carattere usando INVFLG e accettando i movimenti di cursore	$FDF0	—528	COUT1	nessuno
Visualizzazione di un carattere con movimento del cursore	$FDF6	—522	COUTZ	nessuno
Passaggio alla linea seguente (visualizzazione CR)	$FD8E	—626	CROUT	A
Visualizzazione di ERR *via* COUT e emissione di un "beep"	$FF2D	—211	PRERR	A
Emissione di un "beep" da COUT	$FF3A	—198	BELL	A
Posizionamento di BASCL,H in funzione di CV e WNDLFT	$FC22	—990	VTAB	A
Posizionamento di BASL,H in funzione di (A) e WNDLFT	$FC24	—998	VTABZ	A
Posizionamento del cursore all'inizio della linea (fuori dalla finestra)	$FBC1	—1087	BASCALC	A

I tipi di funzionamento dello schermo sono: testo, grafica a bassa risoluzione, grafica ad alta risoluzione, modalità inversa, lampeggiamento.

* MANIPOLAZIONE DEI DATI SULLO SCHERMO

Esistono tre tipi di manipolazione dei dati sullo schermo:

- funzioni grafiche ESC;
- cancellamento di parti dello schermo e spostamento del cursore;
- spostamento delle righe della finestra verso l'alto con soppressione della prima riga e creazione di una nuova riga nella parte inferiore della finestra.

Tabella dei sottoprogrammi di gestione dello schermo

Funzione	Indirizzo esa-decimale	Indirizzo decimale	Nome	Registri distrutti
Rimessa a zero dell'alta risoluzione	$FB33	—1299		A
Passaggio alla prima pagina dello schermo	$FB36	—1226		A
Passaggio in modo testo	$FB39	—1223	SETTXT	A
Caricamento di A=0 per WNDTOP e salto a SETWND	$FB3C	—1220		A
Passaggio in modo grafico	$FB40	—1216	SETGR	A, Y
Passaggio in modo misto testo/grafico	$FB43	—1213		A, Y
CALL-CLRTOP (rimessa a zero della grafica)	$FB46	—1210		A, Y
Caricamento di A=20 per WNDTOP	$FB49	—1270		A
Caricamento di A in WNDTOP	$FB4B	—1205	SETND	A
A=0	$FB4D	—1203		A
Caricamento di A in WNDLEFT	$FBEF	—1201		A
A=40	$FB51	—1199		A
Caricamento di A in WNDWDTH	$FB53	—1197		A
A=24	$FB55	—1195		A
Caricamento di A in WNDBTM	$FB57	—1193		A
A=23	$FB59	—1191		A
Caricamento di A in CV Salto a VTAB	$FB5B	—1189	TABV	A
Passaggio in modo normale	$FE84	—380	SETNORM	Y
Passaggio in modo diverso	$FE80	—384	SETINV	Y
Memorizzazione di Y in INVFLG	$FE86	—374	SETIFLG	nessuno

* FUNZIONI GRAFICHE ESC

Esse sono le seguenti:

ESC	HOME
ESCA	spostamento del cursore verso destra
ESCB	BS
ESCC	LF
ESCD	VP
ESCE	chiamata di CLREOL
ESCF	chiamanta di CLREOL
ESC altro tasto	ritorno al chiamante

Il sottoprogramma che tratta queste funzioni è:

ESC1	(vecchio Monitor)
ESC NEW	(autostart)

* CANCELLAZIONI DI PARTI DELLO SCHERMO E MANIPOLAZIONI DEL CURSORE

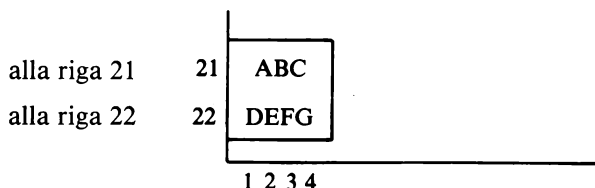
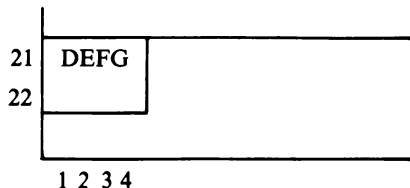
La tabella seguente di indirizzi indica i sottoprogrammi che risultano utili per la cancellazione di alcune parti dello schermo oppure per lo spostamento del cursore.

Funzione	Nome	Indirizzo esa-decimale	Indirizzo decimale	Registri distrutti
Sbiancamento dei caratteri sullo schermo entro la posizione CV, CH fino al fondo della finestra	CLREOF	$FC42	—958	A, Y
Sbiancamento di tutto lo schermo e posizionamento del cursore in alto e a sinistra della finestra (utile in Integer Basic)	HOME	$FC58	—936	A, Y
Sbiancamento dei caratteri situati dopo il cursore nella linea	CLREOL	$FC9C	—868	A, Y
Ritorno all'inizio della linea (CH-0)	CR	$FC62	—926	A, Y
Passaggio alla linea seguente (CV = CV + 1) e, se necessario, chiamata di SCROLL	LF	$FC66	—922	A, Y
Ritorno alla linea precedente (CV = CV — 1)	VP	$FCIA	—998	A
Spostamento del cursore verso destra (CH = CH + 1)	ADVANCE	$FBF4	—1036	A
Spostamento del cursore verso sinistra (CH = CH — 1)	BS	$FC10	—1008	A

* SPOSTAMENTO DELLE RIGHE DELLA FINESTRA DI UNA POSIZIONE VERSO L'ALTO (SCROLL)

Il sottoprogramma SCROLL consente di realizzare questa funzione. Esso si trova all'indirizzo $FC70 (-912).

Esempio:	Se WNDLFT	=1	(colonna di sinistra)
	WNDWDTH	=4	(larghezza)
	WNDTOP	=21	(riga superiore)
	WNDBTM	=23	(riga al di sotto della finestra)

E se abbiamo sullo schermo:

e se chiamiamo SCROLL si otterrà:

* VISUALIZZAZIONE DI UN TESTO ALL'ESTERNO DELLA FINESTRA E UTILIZZO DELLA SECONDA MEMORIA DELLO SCHERMO

In questi due casi, il Monitor non esegue alcun compito. È l'utente stesso che deve inserire i caratteri da visualizzare nelle memorie dello schermo.
Se desiderate avere un testo in una pagina di memoria dello schermo, mentre nell'altra volete avere della grafica a bassa risoluzione, è di gran lunga preferibile riservare la prima pagina per la grafica e la seconda per il testo in quanto il testo è molto più facile da gestire.

* GESTIONE DELLA FUNZIONE GRAFICA A BASSA RISOLUZIONE

Nella funzione grafica standard a bassa risoluzione, il Monitor utilizza le prime 20 righe di testo per 40 righe grafiche lasciando 4 righe di testo nella parte inferiore dello schermo.

Questo è dovuto al fatto che la zona di memoria occupata da un carattere è utilizzata da due punti in bassa risoluzione. Le informazioni relative a questi si trovano su 8 bits e indicano il colore.
La distinzione dei punti viene eseguita mediante il byte MASK.
Il Monitor offre dei sottoprogrammi che consentono:

- la cancellazione della parte grafica dello schermo;
- la scelta dei colori;
- il posizionamento di un punto sullo schermo;
- il tracciamento di righe verticali e orizzontali;
- la determinazione del colore di un punto sullo schermo.

La tabella che riportiamo, fornisce i diversi indirizzi dei sottoprogrammi utilizzabili.

Funzione	Nome	Indirizzo esa-decimale	Indirizzo decimale	Registri distrutti
Visualizzazione sullo schermo di un punto di coordinate ((A), (Y))	PLOT	$F800	—2048	A
Tracciato di una linea orizzontale sullo schermo ((A) = numero linea, (Y) = prima colonna, (44) = ultima colonna)	HLINE	$F819	—2023	A, Y
Tracciato di una linea verticale sullo schermo ((Y) = numero colonna, (A) = prima linea, (45) = ultima linea)	VLINE	$F828	—2008	A
Sbiancamento dello schermo (48 linee grafiche)	CLRSCR	$F832	—1998	A, Y
Sbiancamento della grafica (40 linee grafiche)	CLRTOP	$F836	—1994	A, Y
Posizionamento del colore indicato nell'accumulatore	COLOR	$F85F	—1953	A
Tenuta del colore del punto di coordinate ((A), (Y))	SCRN	$F864	—1948	A = colore

(A) *indica il contenuto del registro* A
(X) *indica il contenuto del registro* X
(Y) *indica il contenuto del registro* Y
(nn) *indica il contenuto dell'indirizzo* nn

Gestione dell'altoparlante tramite il Monitor. Esistono diverse maniere di utilizzare l'altoparlante dell'Apple II.
Quella più utilizzata riguarda la segnalazione degli eventi dei programmi.

Esiste ad esempio un sottoprogramma che consente di far vibrare l'altoparlante alla frequenza di 1 KHz per la durata di 0,1 secondi. Questo sottoprogramma è utilizzato da Reset e tramite il comando CTRL-G.

Tabella indirizzi dei sottoprogrammi monitor che utilizzano l'altoparlante

Funzione	Indirizzo esa-decimale	Indirizzo decimale	Etichetta	Registri distrutti
Se il contenuto dell'accumulatore è $87 (CTRL-G, Bell, codice ASCII = 7), questo sottoprogramma attende 0,01s e emette un "beep"	$FBD9	—1063	BELLI	A, Y
Attesa di 0,01s e emissione di un "beep"	$FBDD	—1059		A, Y
Caricare il registro Y con il valore 192 per 0,1s di "beep"	$FBE2	—1054		A, Y
Fa risonare ("beep") l'altoparlante alla frequenza di 1 kHz per il numero di volte contenuto nel registro Y	$FBE4	—1052	BELL2	A, Y
Fa comparire "ERR" e emette un "beep"	$FF2D	—211	PRERR	A
Emissione di un "beep"	$FF3A	—198	BELL	A

Gestione della cassetta. Esistono due sottoprogrammi principali per la gestione della cassetta che consentono la lettura (READ) e la scrittura (WRITE). Altri sottoprogrammi sono utilizzati internamente dal Monitor. Un file memorizzato sulla cassetta è scomposto in una o più registrazioni. Una registrazione è costituita da una catena di bytes immagazzinati nello stesso tempo partendo da una zona fisica; essa costituisce un'entità fisica sulla cassetta. Un file è costituito da più registrazioni e viene denominato entità logica.
Un programma Basic è costituito da due registrazioni. Studiamone la struttura per i due Basic dell'Apple II.

* INTEGER BASIC

La prima registrazione è costituita solamente da due bytes che servono a memorizzare la lunghezza della seconda registrazione. Questa è utilizzata dal Basic per determinare la fine della zona di memoria occupata dalla seconda registrazione laddove cioè il programma Basic è completo.

* APPLESOFT

In questo caso la prima registrazione è costituita da tre bytes. Come nel caso dell'Integer Basic, i due primi bytes contengono la lunghezza della seconda registrazione; il terzo invece indica la versione dell'Applesoft utilizzata ($55 per la versione standard dell'Applesoft).

Funzione	Etichetta	Indirizzo esadecimale	Indirizzo decimale
Scrive sulla cassetta: - un'intestazione di 10 s - i dati dei quali l'indirizzo d'inizio è contenuto in A1L,H ($3C, 3D) e l'indirizzo finale è contenuto in A2L,H ($3E, $3F) - un byte uguale all'OR ESCLUSIVO di tutti i bytes scritti a titolo di verifica quando si ricollegheranno i dati.	WRITE	$FECD	—307
Legge la cassetta. Mette il primo byte all'indirizzo contenuto in A1L,H e l'ultimo all'indirizzo contenuto in A2L,H. In seguito, legge un byte che deve essere uguale all'OR ESCLUSIVO di tutti i bytes letti. Nei casi contrari ERR è attaccato sullo schermo e CH	READ	$FEFD	—259
Questo programma scrive l'intestazione di sincronizzazione all'inizio di ogni campo di 10 s. È anche usato per indicare l'inizio reale dei dati al momento della lettura.	HEADR	$FCC9	—823
Ricerca di due transizioni sulla cassetta	RD2BIT	$FCFA	—774
Lettura di un bit	RDBIT	$FCFD	—771
Lettura di un byte	RDBYTE	$FCEC	—788
Scrittura di un bit	WRBIT	$FCD6	—810
Scrittura di un byte	WRBYTE	$FEED	—275

Potete lo stesso utilizzare una o più registrazioni per i vostri file. Ogni registrazione è l'entità fisica creata quando viene richiamato il sottoprogramma WRITE. Tale registrazione viene letta mediante la funzione READ. Diamo nella tabella precedente la descrizione dei sottoprogrammi per l'utilizzo della cassetta tramite il Monitor.

Gestione delle manopole di gioco. L'Apple II offre all'utente 4 ingressi digitali, 4 uscite digitali, 4 ingressi analogici e un bit di validazione (vedere Indirizzi speciali. Monopole di gioco). I 4 ingressi analogici sono utilizzati per le manopole di gioco. Il Monitor legge questi ingressi, facendo scattare i contatori e misurando il tempo necessario perché essi ritornino al di sotto di un certo valore. Il sottoprogramma utilizzato è il seguente:

PREAD all'indirizzo $FBIE (-1250)

Il numero (0-3) della manopola (ingresso analogico) deve trovarsi nel registro X prima della chiamata di PREAD; il valore letto si trova al ritorno nel registro Y.
Non devono essere effettuate consecutivamente due letture di ingressi analogici, perché quando il Monitor legge un valore, esso attiva i contatori corrispondenti ai 4 ingressi e questi possono non essere ritornati al valore corretto quando viene richiamato il sottoprogramma PREAD. *Per evitare questo tipo di errore, conviene dunque lasciar passare un certo lasso di tempo tra le due letture.*
Non esistono dei sottoprogrammi di Monitor che utilizzano i 4 ingressi e le 4 uscite digitali accessibili. L'accesso a questi input-output viene fatto utilizzando le istruzioni POKE, PEEK in Basic oppure LDx, STx in assembler. Gli indirizzi corrispondenti vengono indicati al paragrafo *Indirizzi speciali. Manopole di gioco.* Queste uscite sono tuttavia inizializzate all'atto di un'interruzione Reset nella versione autostart del Monitor.
AN0 e AN1 sono inizializzati a 0 (uscite 0 e 1 allo stato TTL basso), AN2 e AN3 sono inizializzati a 1 (uscite 2 e 3 allo stato TTL alto).

Sottoprogramma di attesa WAIT. Il sottoprogramma WAIT è costituito da un insieme di loops nidificati in maniera tale che il tempo d'attesa dipenda dal contenuto dell'accumulatore.
WAIT è situato all'indirizzo $FCA8 (-856).
Esso può essere utilizzato per gestire una frequenza dell'altoparlante oppure una periferica lenta come una stampante o una Teletype.
Il tempo che trascorre mediante la chiamata WAIT, può essere calcolato con la seguente formula:

(2,5A2 + 13,5*A + 13)*1,023 ms.

La tabella che riportiamo a pagina seguente indica alcuni valori ottenuti.

• Gestione delle interruzioni sull'Apple II + .

Generalità. La gestione delle interruzioni è necessaria per permettere di trattare determinati eventi alla loro occorrenza. Nell'intervallo che separa due eventi può essere dunque svolto un programma.
Quando si verifica un'interruzione (evento), il programma in corso viene interrotto, il suo stato viene salvato e un programma specifico viene eseguito. Al termine di quest'ultimo viene restaurato lo stato precedente e il programma interrotto viene rilanciato.
Il microprocessore 6502 consente 3 tipi di interruzione. Esse sono le seguenti:

*RESET
*NMI (interruzione non mascherabile)
*IRQ (e istruzione BRK #).

Contenuto dell'accumulatore	Tempo trascorso in secondi
1	0,000 029
5	0,000 14
9	0,000 34
18	0,001 09
26	0,002 10
32	0,003 07
37	0,004 02
42	0,005 10
46	0,006 06
50	0,007 09
60	0,010 05
75	0,015 43
105	0,029 66
150	0,059 63
205	0,110 32
240	0,150 64
255	0,169 83

Quando si verifica una interruzione, ha luogo il seguente processo:

- se l'interruzione è NMI oppure IRQ (BRK), il contenuto del contatore di programma (PC) e il registro di stato (P) insieme con le altre interruzioni, vengono bloccati;
- altrimenti (RESET) la memoria è messa in lettura finché rimane attivo il sottoprogramma RESET;
- il microprocessore 6502 trasferisce allora al sottoprogramma di gestione il controllo dell'interruzione da trattare utilizzando i vettori di interruzione posti agli indirizzi $FFFA-$FFFF.

Indirizzi dei sottoprogrammi di gestione delle interruzioni

Interruzione	Vettore d'interruzione	Nome	Vecchio monitor	Autostart
NMI	$FFFA-B	NMI	$03FB	$03FB
RESET	$FFFC-D	RESET	$FF59	$FA62
IRQ/BRK	$FFFE-F	IRQ	$FA86	$FA40

Studieremo più precisamente nel seguito di questo paragrafo la gestione dei diversi tipi di interruzione.

Interruzione NMI. Il Monitor dell'Apple II non gestisce queste interruzioni. Esso assicura solamente il trasferimento del controllo dal 6502 all'istruzione situata all'indirizzo $03FB. Dovrete dunque porre a quest'indirizzo un'istruzione JUMP al vostro eventuale programma di trattamento.

Interruzione RESET. Questa ha luogo quando premete il tasto RESET della tastiera o quando mettete in moto il vostro sistema salvo che per tutte le prime versioni. Il suo trattamento si esegue in maniera molto diversa per le due versioni del Monitor.

* IL VECCHIO MONITOR

Quando si verifica un'interruzione RESET, il vecchio Monitor stabilisce la seguente configurazione hardware:
- tastiera come periferica di ingresso dati;
- schermo come periferica di uscita dati;
- tipo di funzionamento normale dello schermo (bianco su sfondo nero), cursore in alto a sinistra (HOME), finestra di testo che occupa tutto lo schermo;
- selezione della prima memoria di schermo.

Il controllo è quindi trasferito all'etichetta MON($FF65), viene emesso un "beep" e il Monitor si appresta a ricevere un comando.

* ROM AUTOSTART

L'interruzione RESET viene qui trattata in diverse fasi:

1. Viene stabilita una configurazione hardware fissa: schermo, tastiera, uscite analogiche, tipo testo, visualizzazione di Apple II.
2. Se il contenuto della memoria (indirizzi da 03F2 a 03F4) indica che si è appena messo in moto il sistema, il Monitor inizializza i vettori di interruzione BRK agli indirizzi da $03F0 a $03F1, di RESET dagli indirizzi $3F2 a $3F3 con il byte di validazione posto all'indirizzo $3F4. Il ruolo di quest'ultimo byte è quello di indicare l'inizializzazione che deve essere fatta: 2. o 3.
Il Monitor autostart ricerca allora una scheda di controllo per dischetti, se essa esiste, nei connettori da 7 a 1. Nel caso favorevole, viene effettuato un salto all'indirizzo Cnoo in cui n è il numero del connettore del controllore, per il caricamento del DOS. Nell'altro caso il comando CTRL-B viene simulato.
3. Se il sistema non deve essere nuovamente inizializzato in maniera completa, il Monitor testa il contenuto del vettore di interruzione (da $3F2 a $3F3).
Se contiene $E000, il Monitor lo posiziona all'indirizzo $E003 (punto di ingresso del CTRL-C) e salta all'inizio del Basic ($E000).
Altrimenti esso salta all'indirizzo indicato.

Test effettuato per sapere quale inizializzazione deve essere operata: questo test si basa sul contenuto del byte di validazione che si trova all'indirizzo

$03F4. *Se, dopo aver eseguito l'*OR *esclusivo fra i valori dei bytes posti agli indirizzi* $3F3 *e* $3F4, *non si riscontra il valore* $A5, *non deve essere effettuata l'inizializzazione completa di cui al punto* 2.

Interruzione IRQ/BRK. Quando si verifica un'interruzione oppure quando viene eseguita un'istruzione BREAK, il microprocessore 6502 segue una sequenza di interruzioni. Questa è la seguente:

- in relazione a IRQ
  * interdizione di altre interruzioni;
  * salto indiretto al sottoprogramma il cui indirizzo si trova a $03FE–$03FF;
  * autorizzazione di interruzione:
- in relazione a BRK

  la sequenza è esattamente la stessa ma si salta all'etichetta BRK invece che al sottoprogramma di trattamento dell'interruzione IRQ.

  Con il vecchio Monitor, l'istruzione che segue BRK viene visualizzata insieme al contenuto dei registri. Con la ROM autostart il Monitor salta all'indirizzo contenuto a $03F0-$03F1. Per défaut questi indirizzi contengono gli indirizzi relativi allo stesso sottoprogramma del vecchio Monitor.

• Utilizzo dei comandi dell'Apple II + come sottoprogrammi

Porre nel buffer di prelevamento dati (GETLN, all'indirizzo $200) i comandi da eseguire, chiamare l'interprete dei comandi del Monitor e tornare indietro. A questo scopo dovrete creare un sottoprogramma assembler che elimina due bytes dalla pila passando in seguito la mano al programma chiamante.
L'esempio che segue indica quello che è necessario fare:

```
10 REM RICHIAMO DEI COMANDI DI MONITOR
20 RM$="2FC:68 68 60 N 2FCG"
30 GOSUB 1000
40 RM$="F800L 2FCG"

50 CALL -936
60 GOSUB 1000
70 END
1000 A=511
1010 L=LEN(RM$)
1020 FOR I=1 TO L
1030 B$=MID$(RM$,I,1)
1040 B=128+ASC(B$)
1050 POKE A+I,B
1060 NEXT
1070 CALL -144
1080 RETURN
```

È possibile accedere ai comandi del Monitor in un altro modo. Dovrete allora creare dei sottoprogrammi assembler che interfaccino il vostro programma con il Monitor.

Indirizzi dei sottoprogrammi corrispondenti ai comandi del Monitor

Indirizzi dei sottoprogrammi corrispondenti ai comandi del Monitor

Sottoprogramma	Indirizzo decimale	Indirizzo esa-decimale	Nome	Registri distrutti
Indirizzo monitor per considerare CR come uno spazio	—512	$FE00	BL1	A, X, Y
Indirizzo monitor per considerare uno spazio	—508	$FE04	BLANK	A, X, Y
Comando per la modifica della memoria	—501	$FE0B	STOR	A
Posizionamento nel monitor quando si incontra + —.	—488	$FE18	SETMODE	A, Y
Posizionamento per un BLANK	—483	$FE1D	SETMDZ	nessuno
Comando di monitor < (registra negli indirizzi A1L,H, A2L,H A4L,H)	—480	$FE20	LT	A, X
Comando MOVE di monitor di A1L,H a A2L,H e verso A4L,H	—468	$FE2C	MOVE	A (Y=0)
Comando di verifica monitor di A1L,H a A2L,H e verso A4L,H	—458	$FE36	VFY	(AY—0)
Comando disassemblatore di 20 istruzioni L	—418	$FESE	LIST	A, X, Y
Posiziona INVFLG in modo inverso $3F	—384	$FE80	SETINV	Y
Posiziona INVFLG in modo normale $FF	—380	$FE84	SETNORM	Y
Posiziona INVFLG con il contenuto di Y	—378	$FE86	SETIFLG	nessuno
Comando O-CTRL-K (ritorno a tastiera)	—375	$FE89	SETKBD	A, X, Y
Scelta dei dati dalla porta il cui numero è nell'accumulatore A2L	—371	$FE8D	INPRT	A, X, Y
Ritorno dallo schermo per l'uscita dati O-CTRL-P	—365	$FE93	SETVID	A, X, Y
Uscita dati dalla porta il cui nu-	—363	$FE95	OUTPORT	A, X, Y
mero è in accumulatore o A2L	—361	$FE97	OUTPRT	A, X, Y
Comando GO scelta indirizzo contenuto in A1 PCL/H	—330	$FEB6	GO	A, X, Y, P
Ripristino dei registri	—327	$FEB9		
Scelta indirizzo contenuto in PCL,H	—324	$FEBC		
Selezione dei registri (CTRL-E)	—321	$FEBF	REGZ	

(segue)

(seguito)

Sottoprogramma	Indirizzo decimale	Indirizzo esa-decimale	Nome	Registri distrutti
Tasto RETURN: - simulazione di uno spazio - scelta puntatore di ingresso al monitor MON Z	—266	$FEF6	CRMON	
Ripristino dei registri STATUS e pila:	—193	$FF3F	RESTORE	
A da ACC	—190	$FF42		
X da XREG	—188	$FF44	RESTR1	
Y da YREG	—186	$FF46		
P da pila e ritorno	—184	$FF48		
Salvataggio dei registri				
ACC a ACC $45	—182	$FF4A	SAVE	
X a XREG $46	—180	$FF4C	SAV1	
Y a YREG $47	—178	$FF4F		
P a STATUS $48	—176	$FF50		
S a SPNT $49	—172	$FF54		
Trasformazione 6502 in modo esadecimale				
Ingresso monitor:				
(autostart) su RESET		$FA62		
su RESET	—167	$FF59	RESET	
Nomi di SETNORM				
Modo schermo normale				
INIT schermo modo testo	—164	$FFSC		
SETVID uscita schermo	—161	$FF5F		
SETKBD ingresso da tastiera	—158	$FF62		
Trasformazione 6502 in modo esadecimale	—155	$FF65	MON	
Emissione di un "beep"	—154	$FF66		
Ingresso monitor	—151	$FF69	MONZ	
Posizionamento di "*" come prompt	—149	$FF6B		
Lettura di una linea	—147	$FF6D		
Inizializzazione del tipo di formato (senza + o —)	—144	$FF70		

(segue)

(seguito)

Sottoprogramma	Indirizzo decimale	Indirizzo esa-decimale	Nome	Registri distrutti
Formato di un comando	—141	$FF73	NXTITM	
- Richiamo di GETNUM per leggere un comando (1 o 2 byte seguiti da una lettera) - Salvataggio della posizione del prossimo carattere in YSAV				
- Richiamo da sottoprogramma al comando desiderato (richiamo di TOSUB)	—126	$FF82		
- Ripristino del carattere da inserire dentro Y e collegato a NXTITM	—123	$FF85		
Lettura di un comando (1 o 2 byte) da inserire in A2L,H e un carattere in cui si inserisce il codice ASCII in accumulatore	—89	$FFA4	GETNOM	
Richiamo di un comando desiderato: - Mettere in pila l'indirizzo del comando - Inserire in accumulatore il modo monitor - RAZ di MODE - RTS (Ritorno al comando)	—66	8$FFBE	TOSUB	
Messa a zero fra due comandi	—57	SFFC7	ZMODE	
Esecuzione dell'istruzione e in (PCL,H) visualizza l'istruzione dei registri ottenuti	—1469	$FA43	STEP	Vecchia versione di monitor
Comando TRACE	—318	$FEC2	TRACE	
Stampa TRACE	—316	$FEC4	STEPZ	

• Sottoprogrammi utilizzabili in linguaggio macchina

Diamo di seguito una lista non limitativa dei sottoprogrammi del Monitor utilizzabili in linguaggio macchina.

Funzioni	Indirizzo decimale	Indirizzo esa-decimale	Nome	Registri distrutti
Inserimento di un CR (ritorno carrello)	—626	$FD8E	CROUT	A
Inserimento di un carattere sullo schermo in posizione CV, CH	—531	$FDED	COUT	A
Inserimento di tre spazi sullo schermo	—1720	$F948	PRBLNK	A, X
Inserimento del numero di spazi (contenuti in X)	—1718	$F94A	PRBL2	A, X
Inserimento dei caratteri in ASCII e in A e (X—1) spazi	—1716	$F94C	PRBL3	A, X
- Emissione di un "beep"	—198	$FF3A	BELL	A
- Inserimento del messaggio "ERR" e di un "beep"	—211	$FF2D	PRERR	A
Inserimento del valore esadecimale della parte bassa dell'accumulatore	—541	$FDE3	PRHEX	A
- Inserimento del contenuto dell'accumulatore	—550	$FDDA	PRBYTE	A
- Inserimento dei registri X, Y	—1728	$F940	PRNTYX	A
- Inserimento dei registri A, X	—1727	$F941	PRNTAX	A
- Inserimento del registro X	—1724	$F944	PRNTX	A
Inserimento di un CR, dei registri X, Y e di un trattino.	—618	$FD96	PRYX2	A, Y
Inserimento dei registri X, Y e di un trattino.	—615	$FD33		A, Y
Inserimento di CR, del valore contenuto in A1H, A1L e di un trattino	—622	$FD93	PRA1	A, X, Y
	—605	$FDA3	XAM8	A (Y=0)
Inserimento dei contenuti esadecimali di (A1L,H e A2L,H)	—589	$FDB3	XAM	A (Y=0)
Salvataggio dei registri A, X, Y, P, S all'indirizzo $45-49	—182	$FF4A	SAVE	A, X
Inserimento di CR, dei nomi dei registri e del loro valore ($45-49X)	—1321	$FAD7	REGDSP	A, X
Inserimento dei nomi e dei valori dei registri	—1318	$FADA	RGDSP1	A, X
Ripristino dei registi A, X, Y, P (non S) dall'indirizzo $45-$49			RESTORE	A, X, Y, P

(segue)

(seguito)

Funzioni	Indirizzo decimale	Indirizzo esa-decimale	Nome	Registri distrutti
Ripristino dei registri (non S) e ritorno all'indirizzo contenuto in A1L,H	—330	$FEB6	GO	A, X, Y, P
Spostamento della zona di memoria compresa tra (A1L,H) e (A2L,H) nella zona di memoria (A4L,H) contenuta in A4	—468	$FE2C	MOVE	A (Y = 0)
Verifica che la zona di memoria all'indirizzo (A4L,H) è identica a quella compresa fra (A1L,H) e (A2L,H). Le differenze sono visualizzate sullo schermo.	—458	$FE36	VFY	A (Y = 0)
Incremento di (A4L,H)	—844	$FCB4	NXTA4	A
Incremento di A1L,H e se A2L,H è inferiore a A1L,H si posiziona il carry	—838	$FCBA	NXTA1	A
Posizionamento di GBASL,H per la linea di cui il numero è nell'accumulatore	—1977	$F847	GBASCALC	A
Se il carry è posizionato RAZ della parte alta dell'accumulatore Se no si scala l'accumulatore verso destra di 4 posizioni (si conserva la parte alta dell'accumulatore)	—1927	$F879	SCRN2	A
Disassemblaggio dell'istruzione all'indirizzo contenuta in PCL,H e visualizzata sullo schermo	—1840	$F8D0	INSTDSP	A, X, Y
Passaggio all'istruzione seguente da disassemblare, aggiunta di LENGTH a PCL,H i risultati sono nei registri X, Y (6502 *deve essere in modo esadecimale*)	—1709	$F953	PCADJ	A, X, Y
Lettura delle paddle	—1250	$FB1E	PREAD	A, Y
Controllo di 0,1 s ed emissione di beep	—1059	$FBDD		A, Y
Carico di Y = 192 per 0,1 s del beep	—1054	$FBE2		A, Y
Emissione di un beep a 1 kHz durante il numero dei cicli indicati in Y	—1052	$FBNE4	BELL2	A, Y

(segue)

(seguito)

Funzioni	Indirizzo decimale	Indirizzo esa-decimale	Nome	Registri distrutti
Posizionamento di un carattere nella memoria schermo se non è un carattere di controllo. Altrimenti trattamento come carattere di controllo	—1027	$FBFD	VIDOUT	A, Y
Pulizia video e posizionamento in alto e a sinistra	—936	$FC58	HOME	A, Y
Messa a 0 di Y e inserimento di un trattino	—612	$FD9C		
Inserimento di un trattino sullo schermo	—610	$FD94		
Inserimento di un carattere sullo schermo come carattere di controllo se (A)<$A0 ritorno a COUTZ.	—528	$FDF0	COUT1	A

Appendici

Appendice 1

Codice ASCII

ASCII Codice	Carattere	ASCII Codice	Carattere	ASCII Codice	Carattere
000	NUL	029	GS	058	:
001	SOH	030	RS	059	;
002	STX	031	US	060	<
003	ETX	032	SPACE	061	=
004	EOT	033	!	062	>
005	ENQ	034	“	063	?
006	ACK	035	#	064	@
007	BEL	036	$	065	A
008	BS	037	%	066	B
009	HT	038	&	067	C
010	LF	039	‘	068	D
011	VT	040	(	069	E
012	FF	041	)	070	F
013	CR	042	*	071	G
014	SO	043	+	072	H
015	SI	044	,	073	I
016	DLE	045	-	074	J
017	DC1	046	.	075	K
018	DC2	047	/	076	L
019	DC3	048	0	077	M
020	DC4	049	1	078	N
021	NAK	050	2	079	O
022	SYN	051	3	080	P
023	ETB	052	4	081	Q
024	CAN	053	5	082	R
025	EM	054	6	083	S
026	SUB	055	7	084	T
027	ESCAPE	056	8	085	U
028	FS	057	9	086	V

(segue)

(seguito)

ASCII Codice	Carattere	ASCII Codice	Carattere	ASCII Codice	Carattere
087	W	101	e	115	s
088	X	102	f	116	t
089	Y	103	g	117	u
090	Z	104	h	118	v
091	[	105	i	119	w
092	\	106	j	120	x
093	]	107	k	121	y
094	↑	108	l	122	z
095	<	109	m	123	↓
096	'	110	n	124	|
097	a	111	o	125	|
098	b	112	p	126	~
099	c	113	q	127	DEL
100	d	114	r		

I codici ASCII sono in decimale
LF = Line Feed (salto linea), FF = Form Feed (salto pagina), CR = Carriage Return (ritorno carrello), DEL = Rubout (cancellazione).

Appendice 2

Codici ASCII prelevabili da tastiera

Tasto	Solo	CTRL	SHIFT	Entrambi	Tasto	Solo	CTRL	SHIFT	Entrambi
spazio	$AØ	$AØ	$AØ	$AØ	RETURN	$8D	$8D	$8D	$8D
Ø	$BØ	$BØ	$BØ	$BØ	G	$C7	$87	$C7	$87
1!	$B1	$B1	$A1	$A1	H	$C8	$88	$C8	$88
2’’	$B2	$B2	$A2	$A2	I	$C9	$89	$C8	$89
3#	$B3	$B3	$A3	$A3	J	$CA	$8A	$CA	$8A
4$	$B4	$B4	$A4	$A4	K	$CB	$8B	$CB	$8B
5%	$B5	$B5	$A5	$A5	L	$CC	$8C	$CC	$8C
6&	$B6	$B6	$A6	$A6	M	$CD	$8D	$DD	$9D
7’	$B7	$B7	$A7	$A7	N̂	$CE	$8E	$DE	$9E
8(	$B8	$B8	$A8	$A8	O	$CF	$8F	$CF	$8F
9)	$B9	$B9	$A9	$A9	P@	$DØ	$9Ø	$CØ	$8Ø
:*	$BA	$BA	$AA	$AA	Q	$D1	$91	$D1	$91
;+	$BB	$BB	$AB	$AB	R	$D2	$92	$D2	$92
,<	$AC	$AC	$BC	$BC	S	$D3	$93	$D3	$93
—=	$AD	$AD	$BD	$BD	T	$D4	$94	$D4	$94
.>	$AE	$AE	$BE	$BE	U	$D5	$95	$D5	$95
/?	$AF	$AF	$BF	$BF	V	$D6	$96	$D6	$96
A	$C1	$81	$C1	$81	W	$D7	$97	$D7	$97
B	$C2	$82	$C2	$82	X	$D8	$98	$D8	$98
C	$C3	$83	$C3	$83	Y	$D9	$99	$D9	$99
D	$C4	$84	$C4	$84	Z	$DA	$9A	$DA	$9A
E	$C5	$85	$C5	$85	→	$88	$88	$88	$88
F	$C6	$86	$C6	$86	←	$95	$95	$95	$95
					ESC	$9B	$9B	$9B	$9B

Appendice 3

Codici ASCII visualizzabili sullo schermo

	Inversa				Flashing				Normale							
									(Controllo)						(Minuscola)	
Esade-cimale	0 $00	16 $10	32 $20	48 $30	64 $40	80 $50	96 $60	112 $70	128 $80	144 $98	169 $AB	176 $B0	192 $C0	208 $D0	224 $E0	240 $F0
0 $0	@	P		0	@	P		0	@	P		0	@	P		0
1 $1	A	Q	!	1	A	Q	!	1	A	Q	!	1	A	Q	!	1
2 $2	B	R	“	2	B	R	“	2	B	R	“	2	B	R	“	2
3 $3	C	S	#	3	C	S	#	3	C	S	#	3	C	S	#	3
4 s4	D	T	$	4	D	T	$	4	D	T	$	4	D	T	$	4
5 $5	E	U	%	5	E	U	%	5	E	U	%	5	E	U	%	5
6 $6	F	V	&	6	F	V	&	6	F	V	&	6	F	V	&	6
7 $7	G	W	‘	7	G	W	‘	7	G	W	‘	7	G	W	‘	7
8 $8	H	X	(	8	H	X	(	8	H	X	(	8	H	X	(	8
9 $9	I	Y	)	9	I	Y	)	9	I	Y	)	9	I	Y	)	9
10 $A	J	Z	*	:	J	Z	*	:	J	Z	*	:	J	Z	*	:
11 $B	K	[	+	;	K	[	+	;	K	[	+	;	K	[	+	;
12 $C	L	\	,	<	L	\	,	<	L	\	,	<	L	\	,	<
13 $D	M	]	—	=	M	]	—	=	M	]	—	=	M	]	—	=
14 $E	N	.	.	>	N	.	.	>	N	.	.	>	N	.	.	>
15 $F	O	-	/	?	O	-	/	?	O	-	/	?	O	-	/	?

Appendice 4

Tabella dei sottoprogrammi assembler utilizzabili, ordinati per indirizzo crescente: Monitor

Indirizzo decimale	Indirizzo esa-decimale	Nome	Funzione	Istruzione APPLESOFT	Pag.
—2048	$F800	PLOT	visualizzazione di un punto a bassa risoluzione	PLOT	
—2023	$F819	HLINE	tracciato di una linea orizzontale	HLIN AT	
—2008	$F828	VLINE	tracciato di una linea verticale	VLIN AT	
—1998	$F832	CLRSCR	sbiancamento dell'intero schermo		
—1994	$F836	CLRTOP	sbiancamento delle linee grafiche a bassa risoluzione (40 1) senza toccare il testo	GR (parte)	
—1953	$F85F	COLOR	posizionamento del colore	COLOR	
—1948	$F864	SCRN	ottenimento del colore	SCRN	
—1728	$F940	PRNTYX	visualizzazione del contenuto dei registri Y e X secondo il formato YYXX		
—1727	$F941	PRNTYA	visualizzazione del contenuto dei registri Y e X secondo il formato AAXX		
—1724	$F944	PRNTX	visualizzazione del contenuto del registro X		

(segue)

(seguito)

Indirizzo decimale	Indirizzo esa-decimale	Nome	Funzione	Istruzione APPLESOFT	Pag.
—1720	$F948	PRBLNK	visualizzazione di tre blanks		
—1718	$F94A	PRBL2	visualizzazione del numero di blanks precisato dal registro X	SPC	
—1716	$F94C	PRBL3	visualizzazione del carattere il cui codice ASCII è nell'accumulatore e di (X)—1 blanks		
—1250	$FB1E	PREAD	lettura della manopola il cui numero è in X	PDL	
—1223	$FB39	SETTXT	passaggio in modo testo	TEXT	
—1216	$FB40	SETGR	passaggio in modo grafico a bassa risoluzione	GR	
—1063	$FBD9	BELL1	attesa di un centesimo di secondo e emissione di un beep, se il contenuto dell'accumulatore vale $87		
—1052	$FBE4	BELL2	emissione, alla frequenza di 1 kHz, di un numero di "beep" uguale al contenuto del registro Y		
—1036	$FBF4	ADVANCE	spostamento del cursore di una posizione verso destra	ESC-A	
—1008	$FC10	BS	spostamento del cursore di una posizione verso sinistra	ESC-B	
—998	$FC1A	UP	ritorno alla linea precedente	ESC-D	
—958	$FC42	CLREOP	sbiancamento dello schermo tra la posizione attuale del cursore e la base dello schermo	ESC-F	
—936	$FC58	HOME	sbiancamento dello schermo e posizionamento del cursore nell'angolo superiore sinistro	HOME ESC-@	
—926	$FC62	CR	ritorno all'inizio della linea seguente		

(segue)

(seguito)

Indirizzo decimale	Indirizzo esa-decimale	Nome	Funzione	Istruzione APPLESOFT	Pag.
—922	$FC66	LF	passaggio alla linea seguente senza cambiare colonna	ESC–C	
—912	$FC70	SCROLL	spostamento delle linee della finestra di una posizione verso l'alto		
—868	$FC9C	CLREOL	sbiancamento dello schermo tra il cursore e la fine della linea	ESC–E	
—756	$FDOC	RDKEY	sostituzione del carattere sotto il cursore dal prossimo carattere acquisito		
—741	$FD1B	KEYIN	lettura di un carattere e visualizzazione		
—715	$FD35	RDCHAR	acquisizione di un carattere con trattamento delle funzioni ESC		
—665	$FD67	GETLNZ	acquisizione di una linea	INPUT	
—550	$FDDA	PRBYTE	visualizzazione del contenuto dell'accumulatore sotto il formato AA		
—531	$FDED	COUT	visualizzazione di un carattere sulla periferica attiva o sullo schermo	PRINT	
	$FDFO	COUT1	visualizzazione di un carattere sullo schermo	PR#0:PRINT	
—384	$FE80	SETINV	schermo in modo diverso	INVERSE	
—380	$FE84	SETNORM	schermo in modo normale	NORMAL	
—336	$FEBO		ritorno al Basic con annientamento del programma	CTRL–B	
—211	$FF2D	PRERR	visualizzazione di "ERR" e emissione di un "beep"		102
—198	$FF3A	BEEP	emissione di un "beep" sulla periferica di uscita attiva		

(segue)

(seguito)

Indirizzo decimale	Indirizzo esa-decimale	Nome	Funzione	Istruzione APPLESOFT	Pag.
—193	$FF3F	RESTORE	ripristino dei registri a partire dagli indirizzi $45-$49		115
—182	$FF4A	SAVE	memorizzazione dei registri agli indirizzi $45-$49		
—151	$FF69	MONZ	input nel monitor		115
—141	$FF73	NXTITM	trattamento di un comando		115

Appendice 5

Tabella degli indirizzi utili nella zona degli input-output

Indirizzo decimale	Indirizzo esadecimale	Significato	
—16384	$C000	input tastiera	
—16368	$C010	tastiera attuata	
—16352	$C020	output cassetta	
—16336	$C030	altoparlante	
—16304	$C050	modo grafico	
—16303	$C051	modo testo	
—16302	$C052	solo grafica	
—16301	$C053	testo e grafica	
—16300	$C054	pagina 1	
—16299	$C055	pagina 2	
—16298	$C056	bassa risoluzione	
—16297	$C057	alta risoluzione	
—16296	$C058	messa a 0 dell'output numerico 0 (TTL)	
—16295	$C059	1	
—16294	$C05A	messa a 0 dell'output numerico 1(TTL)	
—16293	$C05B	1	
—16292	$C05C	messa a 0 dell'output numerico 2 (TTL)	
—16291	$C05D	1	
—16290	$C05E	messa a 0 dell'output numerico 3 (TTL)	
—16289	$C05F	1	
—16288	$C060	input cassetta	
—16287	$C061		0
—16286	$C062	input numerici - pulsante manopola	1
—16285	$C063		2
—16284	$C064		0
—16283	$C065	input analogici - manopole di gioco	1
—16282	$C066		2
—16281	$C067		3
—16272	$C070		

Indice analitico

GUIDA PER L'APPLE

L'APPLE STANDARD

Benoît de Merly

Questo libro si rivolge a tutti coloro che utilizzano un computer Apple e si rivelerà presto indispensabile nella ricerca delle migliori possibilità di questo personal.
Ciascuno vi troverà l'informazione pratica di cui ha bisogno qualunque sia l'applicazione prevista: gestione, calcoli scientifici, giochi, grafica, acquisizione di dati, controllo di processo. Un'opera completa, chiara e pratica che dovrebbe diventare familiare a tutti gli utenti Apple.

L'autore, un ingegnere civile laureato in informatica, ebbe l'idea di scrivere questo libro mentre utilizzava lui stesso un Apple: si è perciò posto le domande che l'utente è portato a porsi se vuol ottenere dal suo computer il massimo delle prestazioni e padroneggiarlo perfettamente.

GUIDA PER L'APPLE

VOLUME 1	VOLUME 2	VOLUME 3
L'APPLE STANDARD	LE ESTENSIONI	LE APPLICAZIONI

I.S.B.N. 88.7688.205.7

Benoît
de Merly

GUIDA PER L'APPLE 1 . L'APPLE STANDARD

E.P.S.I.